VŠB-Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

DIPLOMOVÁ PRÁCE

2013/2014 Bc. Antonín Kučera

VŠB-Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra kybernetiky a biomedicínského inženýrství

Zpracování a analýza zvukového signálu na robotickém zařízení

Acoustic Signal Processing and Analysis for Robotic System

2013/2014 Bc. Antonín Kučera

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání diplomové práce

Student: **Bc. Antonín Kučera**

Studijní program: N2649 Elektrotechnika

Studijní obor: 2601T004 Měřicí a řídicí technika

Téma: Zpracování a analýza zvukového signálu na robotickém zařízení
Acoustic Signal Processing and Analysis for Robotic System

Zásady pro vypracování:

1. Rozbor problematiky algoritmů a metod pro zpracování a analýzu zvukového signálu.
2. Návrh a realizace metod pro analýzu zvukového signálu s detekcí mezí intenzity, hodnot frekvenčního pásma, detekcí jednoduchých povelů.
3. Řešení detekčního a řídicího systému robotického zařízení.
4. Praktické ověření a testování robotického zařízení a jednotlivých implementovaných metod.
5. Zhodnocení dosažených výsledků.

Seznam doporučené odborné literatury:

[1] MANN, Burkhard. *C pro mikrokontroléry : uC & praxe.* 1. české vydání. Praha : BEN-technická literatura, 2003. 279 s. ISBN 80-7300-077-6.
[2] HRBÁČEK, Jiří. *Komunikace mikrokontroléru : s okolím 2.* 1.vyd. Praha : BEN-technická literatura, 2000. 151 s. ISBN 80-86056-73-2.
[3] MAREŠ, Amadeo. *1001 Tipů a triků pro C#.* 1.vyd. Praha : Computer Press, a.s., 2008. 360 s. ISBN 978-80-251-2125-2.
[4] ROZEHNAL, Zdeněk. *Mikrokontroléry MOTOROLA : HC11.* 1.vyd. Praha : BEN-technická literatura, 2001. 191 s. ISBN 80-86056-77-5.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Zdeněk Macháček, Ph.D.**

Datum zadání: 16.11.2012

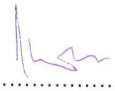
Datum odevzdání: 07.05.2014

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.



.....
Antonín Kučera

Datum odevzdání bakalářské práce: **7.5.2014**

Poděkování

Touto cestou bych chtěl poděkovat vedoucímu mojí diplomové práce panu Ing. Zdeňkovi Macháčkovi, Ph.D. za poskytnutí materiálů, cenných rad a konzultací, které mi ochotně během tvorby diplomové práce poskytl.

Abstrakt

Tato práce spočívá ve vyvinutí programu, který je vložen do procesoru od společnosti Freescale. Úkolem programu je zajistit komunikaci mezi člověkem a robotickým zařízením. Pomocí grafického programu, implementovaného do počítače, jsou zobrazovány jednotlivé odezvy po analýze probíhající v mikroprocesoru, nebo také v samotném programu. Analyzováním a zpracováním zvukového signálu je zařízení schopno rozpoznat různé vstupní požadavky a reagovat na ně, způsobem zobrazení dat na displeji nebo pomocí pulzně šířkové modulace na připojený reproduktor.

Klíčová slova

microprocesor, zvuk, mluvené slovo, analýza dat, rozhodovací algoritmy, reproduktor

Abstract

This work is to develop a program that is implemented in processors from Freescale. The task of the program is to ensure communication between human and robotic equipment. Use a graphics program implemented in the computer are displayed after each response analysis taking place in a microprocessor or also in the program itself. By analyzing and processing the audio signal, the device can detect different input requirements and respond to them accordingly display data on the screen or by using pulse width modulation to the connected speaker.

Keywords

microprocessor, sound, spoken word, data analysis, decision-making algorithms, speaker

Seznam použitých zkratek a symbolů

MCU – mikrokontrolér
DFT - Discrete Fourier Transform
FT – Fourier Transform (Fourierova transformace)
FFT – Fast Fourier Transform (Rychlá Fourierova transformace)
PC – osobní počítač
PCM – pulzně kódová modulace
HMI - Human Machine Interface
ADC – Analog-digital comparator (Analogově digitální převodník)
UD – univerzální deska
TPM – Timer/PWM Module (Modul Časovače/Pulzně-šířkové modulace)
PWM – Pulse-width modulation (Pulzně-šířková modulace)
DSP – Číslicové zpracování signálů
CAN – Controller Area Network (Komunikační sběrnice)
UART - Universal Asynchronous Receiver/Transmitter (sériová asynchronní komunikace)
t – označení času
$\delta_p(t)$ – Diracův impulz
Σ – suma
q – velikost kvantizační úrovně
ε – chyba kvantování
$X(\omega)$ – Fourierova transformace
C_n – koeficient Fourierovy řady
W_N- Činitel s periodou N

Seznam použitých obrázků

Seznam použitých tabulek

Obsah

1. Úvod

Díky stále se rozšiřujícímu trhu s různými druhy mikroprocesorů roste i možnost rozvíjení aplikací a reálného řízení funkčních prvků v elektronice. Za pomoci mnohých periférií procesoru lze využívat různé druhy komunikací a zpracovávat signály. Rovněž můžeme s použitím implementovaného programu se signály pracovat, různě je přetvářet nebo podle nich reagovat a řídit připojené zařízení k MCU. Vytvořením univerzální desky, která má více druhů napájecích napětí o různé intenzitě, zvyšujeme možnost univerzálnosti pro jakékoli aplikace. Použití je reálné od nízkonapěťových systémů, jako jsou např. kamerové čipy, až po různé servomotory nebo jakákoli zařízení, vyžadující vyšší napětí a proud, ale pouze do dvanácti voltů.

Zpracování zvukového signálu zahrnuje mnohé možnosti jak s daným signálem pracovat. Při nahrávání do paměti využívané MCU se považuje za důležité v základním principu, s jakou vzorkovací frekvencí se daný signál nahrává. Metodami pro zpracování signálu je zajištěno, aby byl nahraný vstupní signál před zpracováním co nejkvalitnější a zároveň co nejjednodušší. Kromě metod ukládání, komprimace a komprese dat, využíváme také metody filtrování signálu, změnu hlasitosti vstupního i výstupního signálu, ořezání amplitudy a další. Pomocí algoritmů pro zpracování signálů jsou tyto metody zjednodušeny a jejich zpracování dat tím urychleno tak, aby MCU byl co nejméně zatížen a mohl pracovat přibližně v reálném čase.

Rozborem problematiky algoritmů pro zpracování zvukových signálů je popsána funkčnost dílčích procesů pomocí matematických vyjádření. Teoretickým popisem funkcí je vysvětlen princip funkčnosti a použitelnosti v aplikaci pro zpracování zvuku. Rozdělením signálu je přiblíženo, jaké typy signálů lze v reálných aplikacích zpracovávat. Popisem Fourierovy transformace zobrazujeme jak program vytvořený v mikrokontroléru a aplikaci Visual Studia převádí analogové spektrum signálu na frekvenční. Popisem charakteru řeči je přiblížen náhled odlišností mluvčích, které jsou v analýze zvuku potřebné. Pomocí matematických rovnic a blokových schémat je popsáno jakým způsobem se hledají písmena v mluvených slovech a následně jsou slučovány do slov, ověřovány kde výstupem je textový řetězec. Touto teorií je přiblížen reálný popis programu tvořící automatickou analýzu mluvené řeči. U každého člověka je však problém díky odlišnostem ve vyjadřování, v hloubce hlasu, jazyku národnosti a také na gramatice daného jazyka.

Řešením detekčního zařízení je vytvořena hardwarová část této diplomové práce. Pomocí vyrobené desky plošných spojů se zapojením mikrofonů, přes zesilovače a několik nezbytných elektronických součástek, je měřen vstupní zesílený signál. Tento signál zpracovává univerzální deska s procesorem od společnosti Freescale Semiconductor, MCF51AC256.

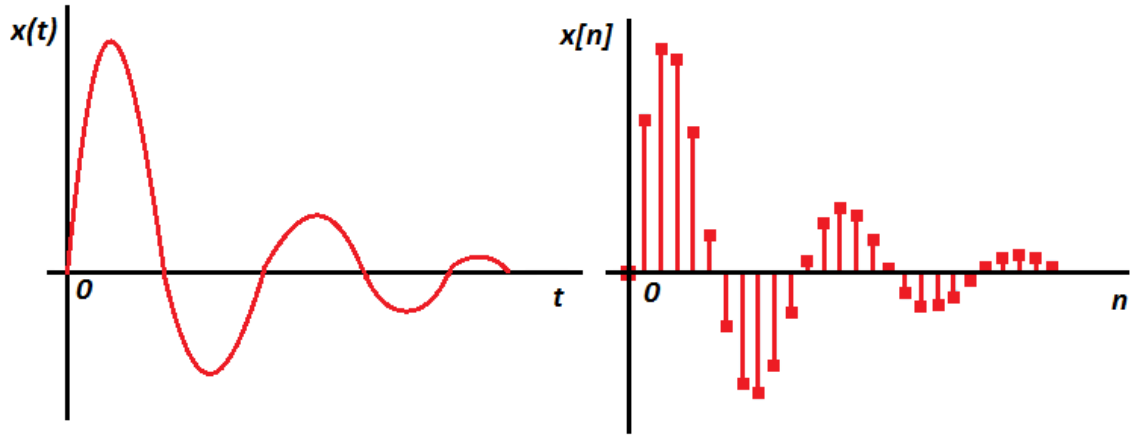
Návrhem a realizací metod pro analýzu zvukového signálu s detekcí mezí intenzity, hodnot frekvenčního pásma, detekcí jednoduchých povelů byl vytvořen program pomocí programovatelného jazyka C a C#. V prvotní verzi spočíval návrh a realizace programu ve vytvoření programu jazykem C#, ale tak aby byl lehce převeditelný do programu v jazyce C. Formou jednoduchých metod s nejjednoduššími funkcemi byl tento program vytvořen a testován tak aby co nejméně zatěžoval MCU.

2. Rozbor problematiky algoritmů a metod pro zpracování a analýzu zvukového signálu.

Každý kmitající předmět lze považovat za zdroj zvuku. Frekvence, jíž kmitá, je vnímána jako výška tónu. Jde o fyzikální jev těles, kdy mechanické vibrace způsobí šíření akustických vibrací, které pak postřehneme jako zmíněný zvukový signál. Když nám těleso kmitá ve správném frekvenčním pásmu, tzn. pro lidské ucho slyšitelné, je tento signál považován za adekvátní pro naše nahrávání a analyzování [1].

2.1. Rozdělení zvukových signálů

Signály pro naši potřebu rozdělujeme dle dvou základních hledisek. Jedná se o spojitost a diskrétnost. Spojité signály jsou takové, jež se vyskytují v analogových obvodech. Tyto signály jsou popsány spojitými funkcemi nezávislých proměnných, které jsou u většiny případů signalizovány jako čas. Diskrétní signály jsou charakterizovány posloupnostmi, jež jsou u nezávislých proměnných reprezentovány celočíselnými hodnotami. Na obrázku uvedeném níže jsou zobrazeny ukázky signálů. V levé části signál spojitý a v pravé části průběh signálu diskrétního.[2]



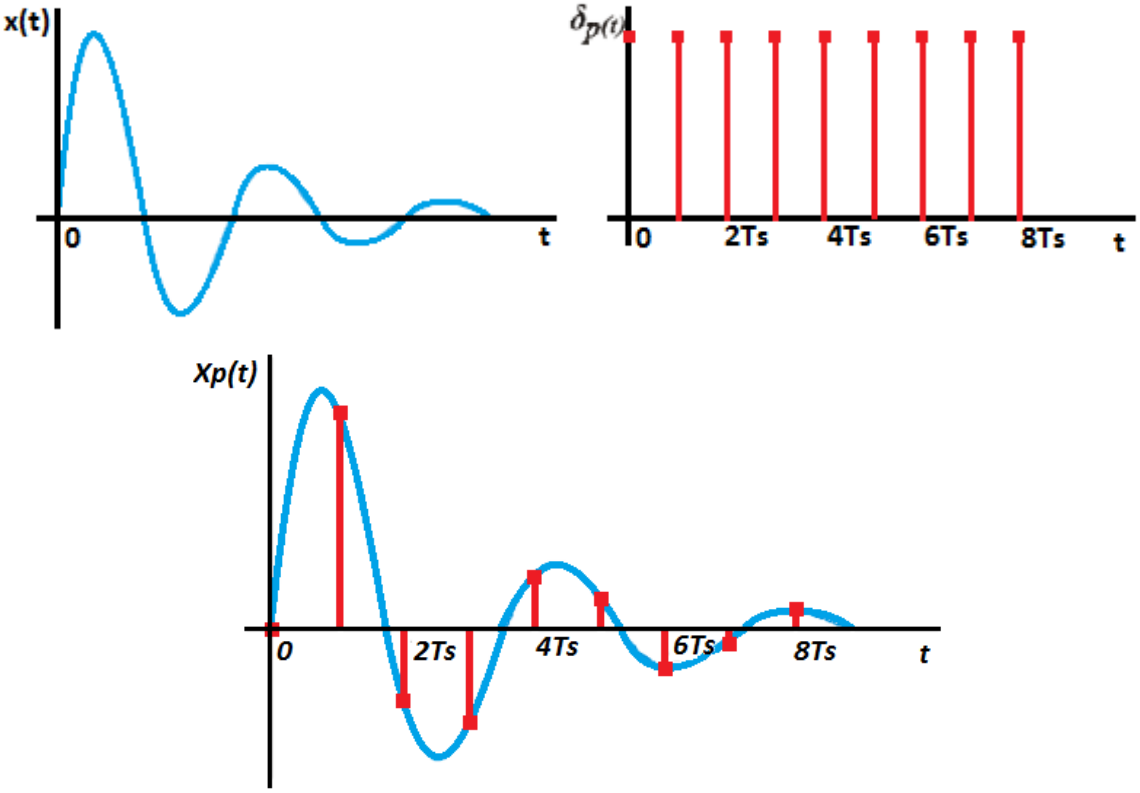
Obrázek 1 - Spojitý a diskrétní signál

2.2. Zpracování signálů

Po nahrání analogového signálu je nutné tento signál převést tak, aby digitální technika byla schopna tento signál dále zpracovávat a analyzovat. Tímto převodem je míněna digitalizace analogového signálu. U MCU převod zajišťuje analogově-digitální převodník, který je jeho součástí jako jedna z periférií. Analogově-digitální převodník při digitalizaci signál musí nejdříve vzorkovat a následně kvantovat.

2.2.1. Vzorkování signálu

Vzorkování, popisované také jako diskretizace, je proces, u něhož se přiřazuje signálu xp(t) ve stanovených časových okamžicích určité funkční hodnoty signálu x(t). Na obrázku níže lze vidět spojitý signál x(t), dále posloupnost Dirackových impulsů δp(t) a výsledný vzorkovaný signál xp(t). Vzorkování signálu se také často označuje jako Shannonův vzorkovací teorém nebo Nyquistův vzorkovací teorém. Rozdíl v těchto názvech je odvozený pouze podle autora. Vzorkovací teorém udává, že spojitý signál může být vzorkován pouze, pokud neobsahuje frekvenční složky nad 0,5 vzorkovací frekvence. [3]



Obrázek 2 - Vzorkovaný spojitý signál

Vzorkování je vyjádřeno jako násobení spojitého signálu posloupností Dirackových impulsů. Výstupní signál je tedy vyjádřen rovnicí (1).

$$x_p(t) = x(t) \cdot \delta_p(t) \tag{1}$$

kde

$$\delta_p(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT_s) \tag{2}$$

platí

$$x_p(t) = \sum_{n=-\infty}^{\infty} x(nT_s)\delta(t - nT_s) \tag{3}$$

2.2.2. Kvantování signálu

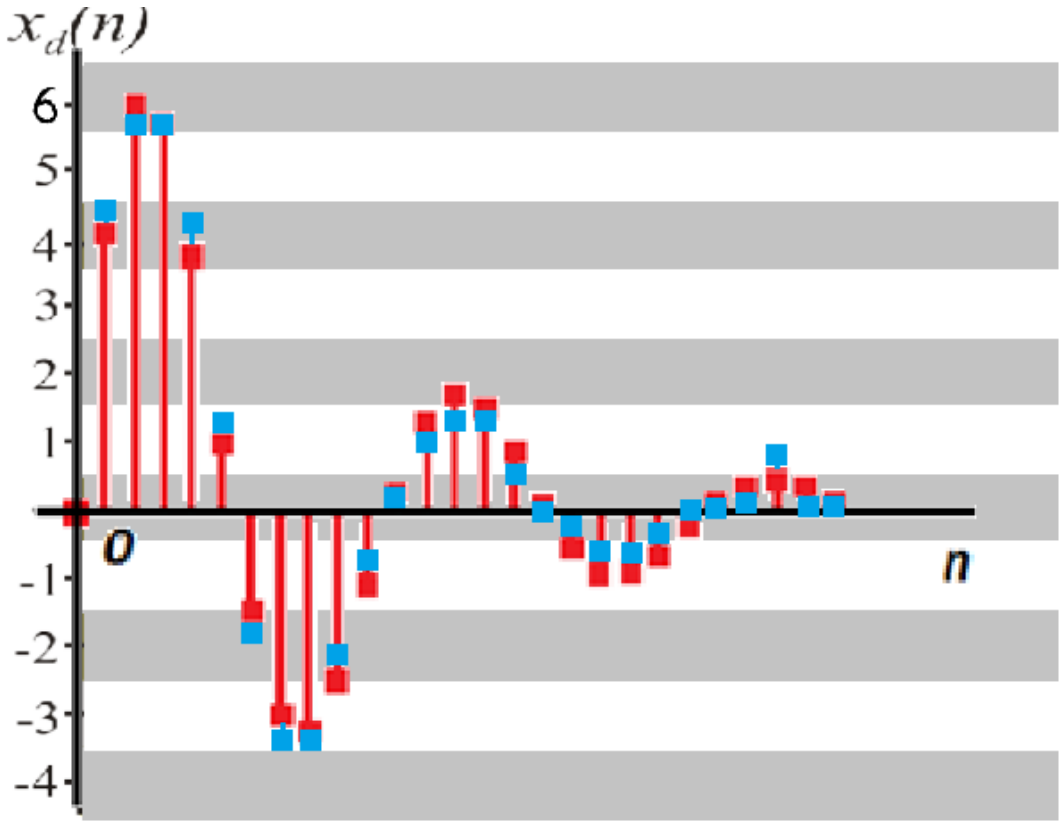
Proces kvantování signálu spočívá v tom, že se převádí spojitý signál na diskrétní. Při transformaci diskrétního signálu na digitální je nutné všechny funkční hodnoty převést na menší počet úrovní. Pro zobrazení digitálního signálu se téměř vždy využívá dvojková soustava. Tím se rozlišovací schopnost převodníku určuje jako N-tá mocnina čísla 2. Velikost kvantizační úrovně q je

$$q = 2^{(-N)} \cdot U \qquad (4)$$

kde U je vstupní rozsah a N počet bitů převodníku. Výsledkem kvanotování je plně digitální signál, lišící se od původního signálu chybou. Její velikost je v případě zaokrouhlování popsána vztahem (5).

$$\varepsilon = \left\langle \frac{-q}{2}; \frac{q}{2} \right) \qquad (5)$$

Signál sestavený z těchto chyb se nazývá kvantizační šum. Na níže uvedeném obrázku je vyobrazeno kvantování signálu. Červené hodnoty ukazují původní signál, modré znázornění označuje kvantovaný signál. Kvantizační intervaly jsou značeny vodorovnými pruhy. [2]



Obrázek 3 - Kvantování signálu

2.3. Fourierova transformace

Jean-Baptiste Joseph baron de Fourier (1768–1830) narozen v rodině krejčího z Auxerre. Po smrti rodičů začal studovat na École Royale Militaire v rodném městě. Nejdříve se angažoval v literatuře, nedlouho poté jej začala zajímat matematika a během svých studii přečetl všechny dostupné práce. Jeho zájem se dále rozvíjet však neupadl, a tak přestoupil do učení bratrů Benediktýnů. Angažoval se také v teoretické fyzice a výzkumu šíření tepla. Svoji vášeň k matematice dále rozvíjel a v revolučním roce odešel vyučovat do řadové (má to tady být řadové nebo řádové?) školy. V roce 1739 byl vlivem událostí zapojen do politické scény, ze které nakonec odstoupil. Působil v oblasti výzkumu šíření tepla. V roce 1807 dokončil svoji práci o šíření tepla v pevných látkách, v níž navrhnul harmonické sinusové signály jako slibný prostředek pro popis šíření tepla tělesem. Díky přednáškám, na nichž přednášel, se Fourier seznámil s množstvím kapacit svého oboru, jako byl Nicolas Carnot, Pierre Laplace nebo Gaspard Monge. Ti v jeho práci zpočátku nevěřili, ale časem se přišlo na to, že Fourier má pravdu a přes to, že sám Fourier mnoho k teorii Fourierových řad nepřinesl, byla významným skokem v celé teorii signálů.[4]

Obrázek 4 – Joseph Fourier

Fourierova transformace je matematická metoda, která je použitelná k analyzování hodnot signálů. V obecném pojetí se jedná o popis funkce, vyjádřený v jiných proměnných pomocí integrální transformace. Ve speciálním případě se uvažuje tzv. trigonometrická Fourierova transformace, která za bázové funkce pokládá sin(kt), cos(kt) nebo v komplexním tvaru exp(ıkt), kde k je celé číslo v případě Fourierovy řady nebo reálná proměnná v případě Fourierovy transformace.[5]

2.3.1. Princip Fourierovy transformace

Fourierova transformace je vyjádřením časově závislého signálu pomocí harmonických signálů, funkcí sin a cos, obecně tedy funkce komplexní exponenciály. Slouží pro převod signálů z časové oblasti do oblasti frekvenční. Signál může být buď ve spojitém, nebo v diskrétním čase [6]

Fourierova transformace $X(\omega)$ z funkce $x(t)$ je definována integrálním vztahem,

$$X(\omega) = \int_{-\infty}^{\infty} s(t) e^{-j\omega t} dt \qquad (6)$$

který existuje, je-li funkce *x(t)* absolutně integrovatelná:

$$\int_{-\infty}^{\infty}|x(t)|dt < \infty \qquad (7)$$

Funkci *x(t)* vypočteme z *X(ω)*inverzní Fourierovou transformací:

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} S(\omega)e^{j\omega t}d\omega \qquad (8)$$

2.3.2. Fourierova řada periodické funkce

Nejjednodušší odvození FT vychází z Fourierovy řady periodické funkce (9), jejíž funkci lze nalézt ve skládání dvou kmitů stejného směru s takovými frekvencemi, aby výsledná funkce mohla být periodická, tedy $T_l = nT_n$, kde *n* je celé číslo. Funkce daná touto superpozicí bude mít tvar

$$f(t) = B_0 + \sum_{n=1}^{\infty} A_n sin(n\omega_1 t) + \sum_{n=+}^{\infty} B_n cos(n\omega_1 t) \qquad (9)$$

An, Bn jsou koeficienty tvořící tzv. spektrum funkce f(·).

Koeficienty Fourierovy řady

Nejprve budeme uvažovat, že funkce je periodická na intervalu <0, T1> a předpokládat platnost výše uvedeného rozvoje. Obě strany rovnosti vynásobíme funkcí sin(mω1t) a prointegrujeme přes interval délky $T_1 = \frac{2\pi}{\omega 1}$. Dostaneme rovnici

$$\int_0^{T_1} f(t)sin(m\omega_1 t)dt = B_0\int_0^{T_1} sin(m\omega_1 t)dt + \sum_{n=1}^{\infty} A_n \int_0^{T_1} sin(m\omega_1 t)sin(n\omega_1 t)dt + \sum_{n=1}^{\infty} B_n \int_0^{T_1} sin(m\omega_1 t)cos(n\omega_1 t)dt \qquad (10)$$

a využitím kolmosti funkcí 1, sin, cos dostaneme

$$\int_0^{T_1} f(t)\, sin(m\omega_1 t)\, dt = \frac{A_m T_1}{2} \qquad (11)$$

Fourierovu řadu můžeme také vyjádřit v komplexním tvaru. Když vezmeme v úvahu Eulerovy vztahy pro funkce cos, sin, exp jako

$$f(t) = \sum_{n=-\infty}^{\infty} C_n exp\,(in\omega_1 t), \qquad (12)$$

kde pro koeficienty C_n platí vztah

$$C_n = \frac{2}{T_1}\int_0^{T_1} f(t)\, exp(-in\omega_1 t)\, dt. \qquad (13)$$

2.3.3. Diskrétní Fourierova transformace

Tato transformace se používá v případě zpracování číslicového signálu. Pracujeme s konečným počtem hodnot a to jak v časové, tak frekvenční oblasti. Signál má stejný počet hodnot N v obou případech. V anglofonním prostředí se transformace označuje jako DFT – Discrete Fourier Transform.[5] Zahrnuje komplexní řadu x[n] o N prvcích

$$x_0, x_1, x_2, \dots, x_{n-1} \tag{14}$$

kde každé *x* je komplexní číslo.

$$x_i = x_{Re} + jx_{Im} \tag{15}$$

Za předpokladu, že řada vně rozsahu 0 až N-1 je periodické prodloužení základního intervalu, pak můžeme DFT definovat jako

$$X[k] = \sum_{k=0}^{N-1} x[n] e^{-jn\theta_k} = \sum_{n=0}^{N-1} x[n] e^{\frac{-j2\pi nk}{N}}, k = 0,1, \dots N-1 \tag{16}$$

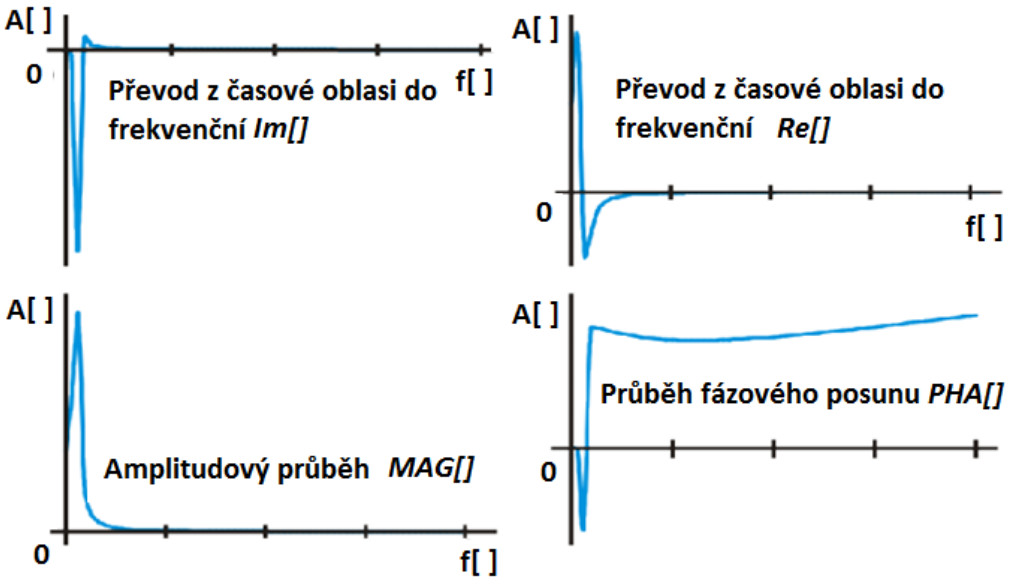
pak se zpětná transformace charakterizuje jako

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{jn\theta_k} = \frac{1}{N} \sum_{n=0}^{N-1} X[k] e^{\frac{j2\pi k}{N}}, k = 0,1, \dots N-1 \tag{17}$$

Zde popsaná transformace pracuje se vstupní komplexní řadou. Můžeme vynulovat imaginární část, naplnit reálnou část, a tudíž pracovat jen s reálným signálem. Výsledný signál je opět komplexní řadou a dělí informaci na dvě části, které však bez převedení do polárních souřadnic nedávají smysl. Můžeme vypočítat amplitudu fázového posunu dle známých vztahů.[5]

$$MAG[k] = \sqrt{ReX[k]^2 + ImX[k]^2} \tag{18}$$

$$PHA[k] = arctan\left(\frac{ImX[k]}{ReX[k]}\right) \tag{19}$$



Obrázek 5 - Příklad DFT

2.3.4. Rychlá Fourierova transformace

V roce 1965 zveřejnili J. W. Cooley a J. W. Tuckey ve své práci *Algoritmus pro strojní výpočet komplexní Fourierovy řady* popis výpočtu rychlé Fourierovy transformace (FFT). Ve skutečnosti byla tato technika objevena o několik desetiletí dříve. Německý matematik K. F. Gauss ji popsal přibližně o sto let dříve. Jeho práce však upadla v zapomnění, protože v jeho době neexistovala hlavní věc, pro kterou byla FFT navržena - digitální počítač.[6]

Největší výhodou FFT je značná rychlost oproti DFT. Pro výpočet *N* hodnot pomocí DFT(11) je potřeba provést N^2 komplexních násobení a *N(N-1)* komplexních sčítání. Doba pro výpočet je časově náročná a neefektivní. Naproti tomu doba výpočtu FFT je $N/2 \cdot log_2(N)$.[7]

Algoritmus rychlé Fourierovy transformace

Vztah pro výpočet diskrétní posloupnosti *X(k)* ze zadané posloupnosti hodnot *x(n)* je dán vztahem (16). S ohledem na výpočty předešlých rovnic zavedeme substituci

$$W_N{}^{nk} = e^{\frac{-j2\pi nk}{N}} \qquad (20)$$

Vztahy pro DFT a I DFT pak budou ve tvaru

$$DFT[x(n)] = X(k) = \sum_{n=0}^{N-1} x(n) e^{\frac{-j2\pi nk}{N}} = \sum_{n=0}^{N-1} x(n) W_N{}^{nk} \qquad (21)$$

$$IDFT[X(k)] = x(n) = \frac{1}{N}\sum_{k=0}^{N-1} K(k) e^{\frac{j2\pi nk}{N}} = \frac{1}{N}\sum_{k=0}^{N-1} X(k) W_N{}^{-nk} \qquad (22)$$

Výpočet diskrétních hodnot X(k) nebo x(n) podle uvedených vztahů je při delší posloupnosti N zadaných hodnot časově velmi náročný kvůli velkému množství aritmetických operací. Počet výpočetních operací můžeme snížit použitím vhodného algoritmu, který využívá periodicity činitele WNnk. Z exponenciálního vyjádření rovnice (20) vyplývají následující vlastnosti:

Činitel $W_N{}^{nk}$ vykazuje periodicitu s periodou N, takže platí:

$$W_N{}^{nk} = W_N{}^{nk+N} = W_N{}^{nk+2N} = \cdots \qquad (23)$$

Pro *N/2* – bodovou a *N*-bodovou DFT bude:

$$\mathrm{W_{N/2}}^{\mathrm{nk}} = \mathrm{W_N}^{\mathrm{2nk}} \qquad (24)$$

Činitel $W_N{}^{nk}$ se vyznačuje souměrností, neboť platí:

$$W_N{}^{nk+N/2} = exp\left[\frac{-j2\pi nk}{N}\right] exp\left[-j\frac{2\pi}{N}\frac{N}{2}\right] = -W_N{}^{nk} \qquad (25)$$

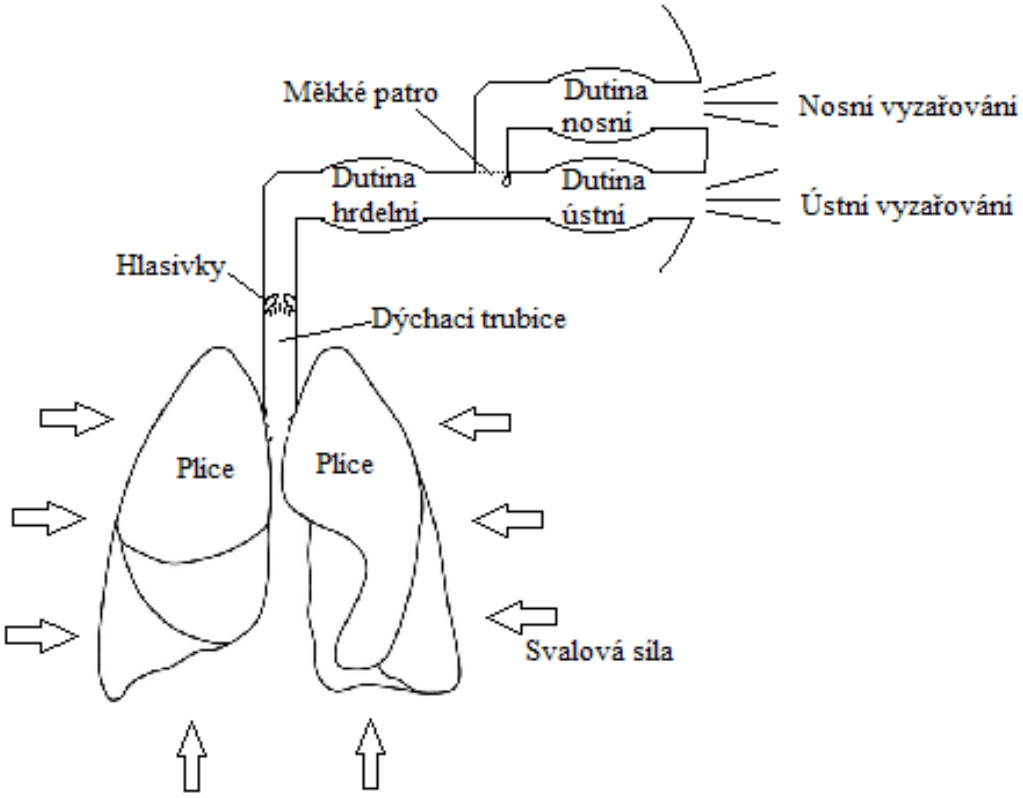
Rozšířeným algoritmem pro výpočet hodnot *X(k)* nebo *x(n)*, který účinně využívá uvedených vlastností, je rychlá Fourierova transformace. Základní myšlenka FFT vychází z rozkladu zadané posloupnosti *N* diskrétních hodnot na kratší posloupnosti, jejichž DFT vyžaduje menší počet aritmetických operací. Následující kombinací dílčích výsledků lze získat DFT původní delší posloupnosti při podstatně menším celkovém počtu aritmetických operací. Rozklad na kratší posloupnosti lze provést např. rozdělením zadané posloupnosti *N* hodnot na dvě části a to na posloupnost sudých a lichých členů.[5]

2.4. Charakter řečového signálu

Každý člověk má charakteristický styl mluvení. Hlavními odlišnostmi jsou barva, výška a hloubka hlasu. Proces tvorby řečových signálů je však u všech stejná.

2.4.1. Proces tvorby řeči člověkem

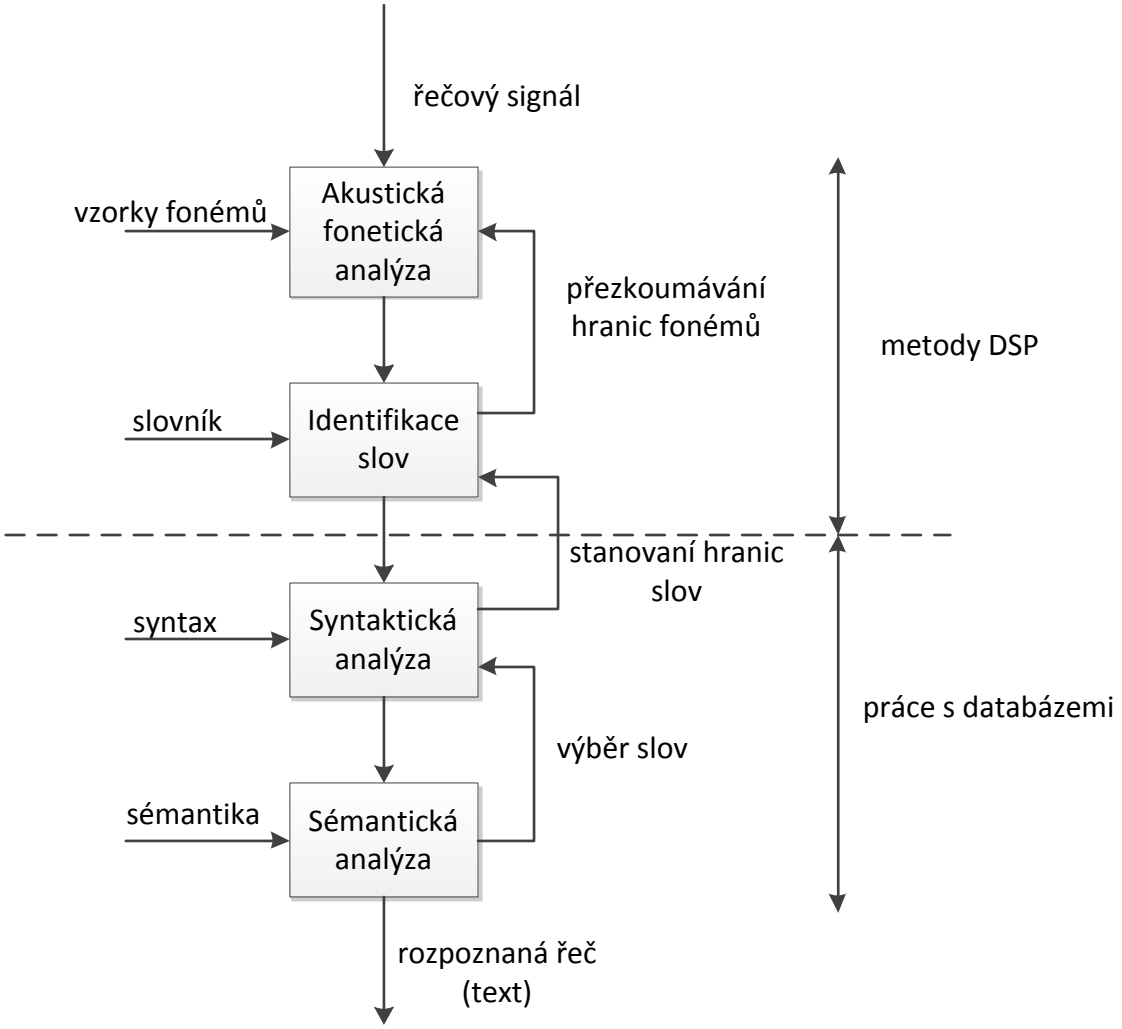
Princip tvorby řeči je velice podobný jako u vytváření zvukových tónů u dechových hudebních nástrojů. Hlavním faktorem celého procesu je proud vzduchu vydávaný plícemi nebo zařízením vytvářející tlak hnaný do nástroje, jenž dále prochází úzkou mezerou, tzv. hlasivkovou štěrbinou. Tu obklopují hlasivky, které se rozkmitají, a tím mění proud vzduchu na pravidelný signál. Při kmitání hlasivek vznikají vzduchové rázy přibližně v intervalech 10ms. Vytvořené signály, jež ještě nepovažujeme za řečové, jsou zpracovány v hlasovém traktu, do nějž spadá dutina nosní, hrdelní a ústní, pomocí pohybu řečových orgánů. Výsledkem artikulace jsou různé opakovatelné zvuky nazývající se hlásky. Tlak vzduchu se šíří kulově, proto energie signálu, přicházející k posluchači, velmi rychle klesá. Schematicky zjednodušený fyziologický model hlasového ústrojí, podílejícího se na vytváření řeči, je zobrazen na obrázku níže.[9]



Obrázek 6 - Schématické zobrazení lidského hlasového ústrojí

2.4.2. Princip automatického rozpoznávání řeči

Uplatnění jednotlivých lingvistických nauk v procesu rozpoznávání řeči, jako jsou fonetika, fonologie, prozodie, lexikologie, syntax a sémantika, zobrazuje blokové schéma na obrázku níže. U vstupního signálu musí být nejprve rozpoznány jednotlivé fonémy pomocí fonetické anýzy. V následujícím bloku z časové posloupnosti jsou vytvořena slova rozpoznaných fonémů, kterou jsou porovnány se slovy v již uloženém slovníku. Pokud nejsou daná slova nalezena, opakuje se proces s přezkoumáním hranic původních fonémů. Tato fáze se provádí pomocí metod pro číslicové zpracování signálů. Pomocí syntaktické analýzy v následujícím bloku se stanovují hranice slov pro správné řazení do slovních spojení. Tento proces spočívá v práci s objemnými databázemi, jež obsahují kompletní gramatiku analyzovaného jazyka. V posledním bloku dochází k přezkoumání, jestli slovní spojení má smysluplný význam. Výstupem ze systému je textový řetězec obsahující převedenou řeč. [9]



Obrázek 7 - Blokové schéma principu rozpoznávání řeči

2.4.3. Informační obsah řeči

Řeč reprezentuje informační obsah nebo fyzikální řečový signál. Pro automatické rozpoznávání řeči se ukazuje jako zásadní to, zda se vyskytuje ve formě řečového signálu. Řeč inteligentního člověka je charakteristická několika strukturami. Akustická struktura je závislá na amplitudě a frekvenci řeči. Lingvistická struktura znamená styl skladby a gramatiky mluvení. Subjektivní vliv osobnosti ovlivňuje intonaci, rytmus, barvu hlasu atd.
Řečový signál je nositelem těchto informací:

- vlastní tělo zprávy obsahující písmena a slova
- identitu mluvícího člověka
- druhem sdělení – jakým stylem je řečeno, zda je zesílené, zdůrazněné apod.
- nálada mluvčího – projevuje se na stylu promluvy
- původ mluvčího – jazyk, nářečí, přízvuk
- charakteristika prostředí, ve kterém je mluveno

Při automatickém rozpoznávání jsou zpracovávány první dvě kategorie informací. Člověk na druhou stranu dokáže současně zpracovat všechny typy informací.

Reprezentace řeči

Reprezentace řeči jejím informačním obsahem ve vztahu ke sdělení se používá k výpočtům v oblasti teorie informace zavedena Shannonem [10]. Řeč se skládá z konečného počtu fonémů. Informační obsah znaku x_n z množiny $x_1, x_2, \ldots x_N$ je definován jako logaritmus převrácené hodnoty pravděpodobnosti, se kterou se znak vyskytuje

$$I(x_n) = log_z \frac{1}{P(x_n)} = -log_z P(x_n) \qquad (26)$$

kde abeceda znaků x_n tvoří úplnou soustavu, pro kterou platí

$$\sum_{n=1}^{N} P(x_n) = 1 \qquad (27)$$

Je-li zvolen algoritmus o základu $z=2$, je pak jednotka obsahu řeči jeden bit. Další důležitá veličina je průměrný informační obsah jednoho znaku. Tento znak představuje entropii zdroje

$$H = -\sum_{n=1}^{N} P(x_n) log_z P(x_n) \qquad (28)$$

Předpokládáme, že jsou všechny znaky staticky nezávislé. Jestli se všechny znaky vyskytují se stejnou pravděpodobností, jedná se o maximální entropii. Odchylka entropie H od maximální hodnoty se nazývá nadbytečnost zdroje

$$R = H_{max} - H \qquad (29)$$

tato nadbytečnost vzniká neoptimálním kódováním znaků, nebo jejich statickou závislostí.

Relativní četnost fonologických jednotek mluvené češtiny

Hrubý odhad obsahu řeči vychází z dvou dílčích pozorování. Předpokládejme průměrně 40 fonémů k popisu jazyka. Pokud jsou jednotlivé hlásky statisticky nezávislé a mají stejnou pravděpodobnost, vychází entropie $H_{max} \approx 5$ bitů na hlásku. Při obvyklé řeči je vysloveno přibližně 10 hlásek za sekundu. V tom případě nevychází informační rychlost vyšší než 50 bitů/s. Ve skutečnosti je ale tato hodnota menší, protože se hlásky nevyskytují se stejnou pravděpodobností a existuje také statická vazba mezi po sobě jdoucími hláskami. Četnost výskytů hlásek v mluvené češtině je vypsána v tabulce níže. Reálná hodnota entropie tedy klesne na *H=3-3,5* bitů, které odpovídá rychlost přenosu informace na 35 bitů/s. Podle výsledků psychoakustiky bylo zjištěno, že člověk je schopen zpracovat maximálně informace o rychlosti 50 bitů/s.[11]

Foném (hláska)	Relativní četnost výskytu [%]	Foném (hláska)	Relativní četnost výskytu [%]	Foném (hláska)	Relativní četnost výskytu [%]
Pauza	16,6	R	3,02	Ř	1,01
E	8,16	P	2,59	CH	1,00
O	5,75	J	2,41	Č	0,80
A	5,56	U	2,30	OU	0,73
I	5,01	D	2,28	Ť	0,71
T	3,97	Á	2,00	Ž	0,70
S	3,92	Ň	1,67	F	0,61
N	3,80	Z	1,62	Ú	0,49
L	3,62	B	1,53	Ď	0,40
Í	3,36	C	1,15	G	0,34
K	3,35	H	1,06	Ó	0,06
V	3,24	É	1,04	AU	0,02
M	3,09	Š	1,03	EU	0,00
Σ	**69,43**	**Σ**	**23,70**	**Σ**	**6,87**

Tabulka 1 - Relativní četnost výskytu fonologických jednotek mluvené češtiny

Navzorkovaný řečový signál vykazuje oproti předcházející úvaze mnohem vyšší informační rychlost. Např. u vzorkování 8 kHz a kódování vzorků 8 bity je informační rychlost 64000 bitů za sekundu.

Díky těmto údajům je zřejmé, že se v signálu objevuje veliká redundance (barva hlasu, šum okolí, defekty řeči, atd.). Díky vysoké redundanci je zajištěna dobrá spolehlivost komunikace. Řeč zůstává srozumitelná, i když je signál rušen nebo jakkoli omezován. Kdyby se ale podařilo redundanci odstranit, případně lépe potlačit, bylo by možné lépe přenášet a zaznamenávat řečový signál.

3. Řešení detekčního a řídicího systému robotického zařízení.

Jako řešení detekčního systému byla navržena univerzální deska s MCU označením MCF51AC256. Tato deska je univerzální s vyvedenými komunikačními periferiemi a nepoužité vstupy a výstupy jsou vyvedeny na svorkovnici, ke které lze připojit libovolné zařízení v rámci povolených funkcí MC. Analýza zvuku využívá jen dva vstupní a jeden výstupní pin na procesoru. Na univerzální konektor je přepojena deska plošných spojů se zapojením zesilujících prvků. Ty slouží pro zesílení vstupního signálu.

3.1. Mikrokontrolér MCF51AC256

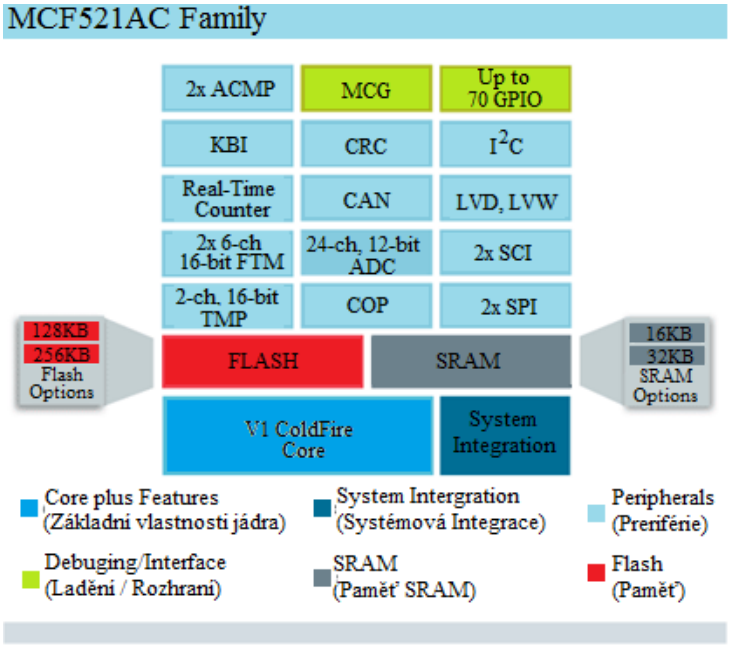
Vzhledem k nasycenosti trhu různými typy MCU s různými vlastnostmi a požadavky se stále vyvíjí nové druhy hlavně s vyšším výkonem a dalšími vylepšeními. S vyšším výkonem však přichází větší rušení a problém s jeho odstíněním. Společnost Freescale reagovala na tento problém a představila novou rodinu mikrokontrolérů s označením Flexis AC. Tato skupina obsahuje 8bitové kontrolery s typovým označením MC9S08AC128/96/60/48/32 a 32bitové V1 ColdFire mikrokontroléry MCF51AC256/128. Rodina Flexis je navržena přímo do rušivých prostředí. Je vhodná pro použití jednočipových aplikací u řízení motorů a HMI řešení (Human Machine Interface), zahrnujících bílé spotřebiče, jako jsou pračky, myčky nádobí, ledničky apod.[12]

Obrázek 8 - Vzhled MCF51AC256

Základní vlastnosti MCF51AC256 - 32-bit ColdFire:

- 50MHz 32bitové jádro ColdFire V1 s 25MHz sběrnicí
- Až 32kB SRAM/až 256 kB Flash
- 2x 6kanálový, 16bitový modul FlexTimer (FTM), 1x 2kanálový 16bitový TPM
- 24kanálový 12bitový ADC
- msCAN rozhraní
- 2x SPI, 2x SCI a I2C
- Multi-clock generator
- Až 70 GPIO

Grafické vyjádření základních vlastností MCU



Obrázek 9 - základní vlastnosti MCU

Dostupné periferie:

ACMP – Tato periferie obsahuje dva nezávislé analogové komparátory. Tento obvod poskytuje porovnávání dvou vstupních napětí nebo jednoho s vnitřním referenčním napětím. Porovnávací obvod může pracovat v plném rozsahu napájecího napětí.

KBI – Periferie klávesového přerušení obsahuje až osm nezávislých externích zdrojů přerušení.

CRC – Generátor cyklické redundantní kontroly poskytuje kód pro detekci chyb a jednoduchou kontrolu na všech dostupných paměťových místech ať už v paměti FLASH, nebo RAM.

I^2C – Poskytuje způsoby komunikace mezi několika zařízeními. Rozhraní je určeno pro komunikaci až do 100 kb/s.

Real-Time Counter – Je čítač hodinového pulzu, který bez závislosti na programu procesoru určuje časovou konstantu v reálném čase.

CAN – Je komunikační sběrnice původně vyvinuta pro automobilový průmysl. Je to sériová datová komunikace pro vozidla navržená tak aby odolávala všem rušením a elektromagnetickým vlivům.

LVD, LVW – Mikrokontrolérové řady MFC51AC256 zahrnují systém pro ochranu obsahu v paměti proti podmínkám nízkého napětí a kontroluje MCU stavy systému při kolísání napájecího napětí.

FTM – Je šesti-kanálový čítač, který podporuje vstupní snímání, výstupní porovnávání a generování PWM signálů pro ovládání elektromotorů a správu napájení aplikací. FTM je zpětně kompatibilní s TPM pro jednoduchou konfiguraci a provoz s vysokou mírou přizpůsobivosti.

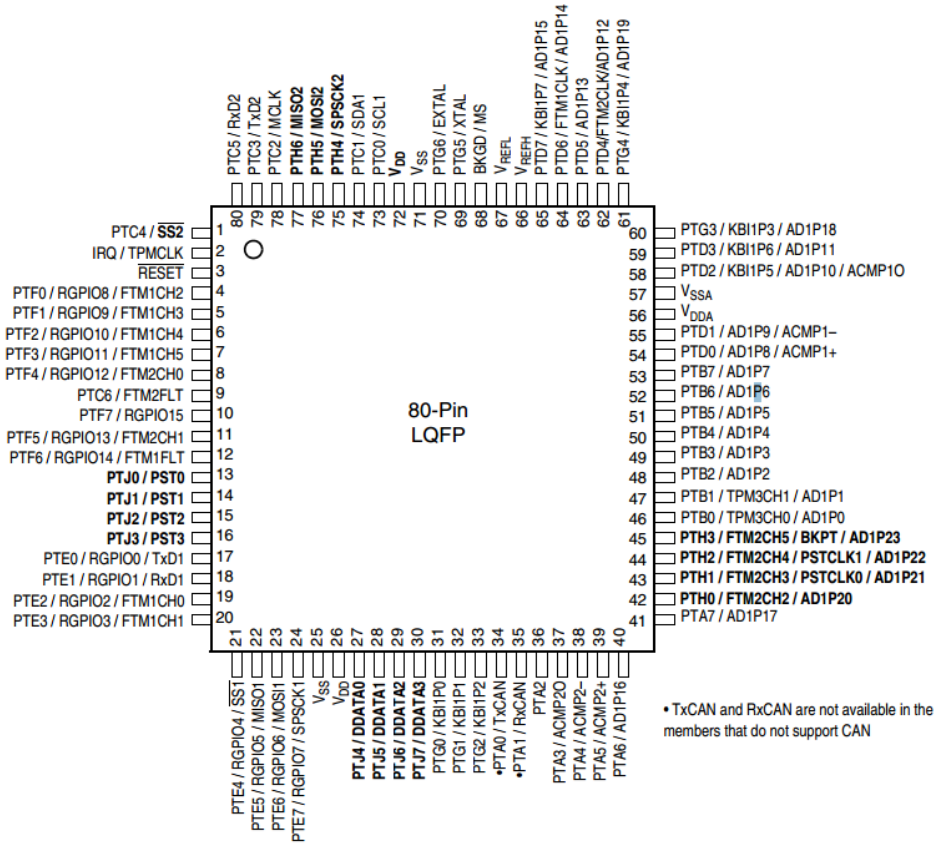
ADC – Periferie 12bitového analogově-digitálního převodníku s postupnou aproximací je konstruován pro provoz v rámci integrovaného systému na čipu mikrokontroléru. Jeho funkcí je převádět analogový signál přicházející na vstup převodníku do digitální podoby pro následující zpracování.

SCI – Sériové komunikační rozhraní pracující v asynchronním režimu např. pro linky RS232 respektive 485

SPI – Sériové 8bitové komunikační rozhraní stejné jako SCI s tím rozdílem, že pracuje v synchronním režimu. Stejně jako SCI bývá tento modul často reprezentován integrovaným obvodem USART 8251. Dále taky bývají integrovány to velké řady mikrokontrolérů hlavně do PICmicro MCU a do MCU od společnosti Atmel.

TMP – pouze jedno až osmikanálový systémový časovač, který podporuje standardní zachycení vstupní a výstupní porovnávané hodnoty, nebo náběžnou hranu PWM na každém kanálu. Kontrolní bit umožnuje, aby TPM byl nakonfigurován tak, že všechny kanály mohou být použity pro zarovnání na střed PWM funkcí. Časovací funkce jsou založeny na 16bitovém čítači s děličem a modulem funkcí pro ovládání frekvence a rozsah časové reference. Tento časový systém je ideální pro řízení široké škály aplikací. Díky zarovnanému PWM na střed je ovládání vhodné pro řízení motorů a dalších nízkonapěťových systémů.

Grafický nákres rozvržení 80pinového MCU[12]



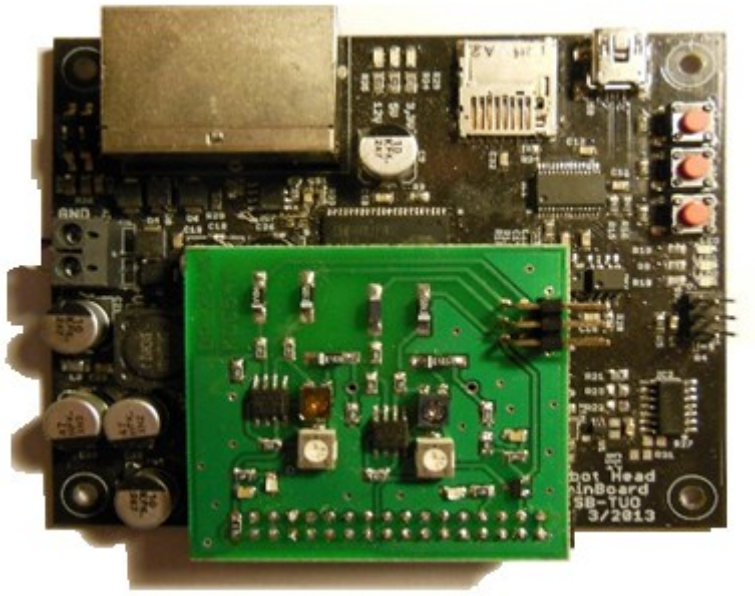
Obrázek 10 - rozložení periferií na procesoru MCF51AC256 - 80-pin

3.1.1. Využití MCU v analýze zvuku

Použitý MCU je umístěn na univerzální desce, která má vyvedeny všechny periferie procesoru. Komunikační protokoly jsou vyvedeny na samostatném konektoru. Ostatní vstupy a výstupy MCU jsou vyvedeny společným konektorem, ke kterému je možno připojit jakékoli rozšíření. Pro účely analýzy zvuku je tento typ MCU ideálním z hlediska odolnosti kvůli rušení okolními vlivy. Díky vestavěnému ADC převodníku není potřeba navrhovat a vyrábět externí ADC. Tím se v první řadě zrychlí funkce převodu analogového signálu a také ukládaný signál v procesoru nezíská žádné zkreslující hodnoty (rušení), které by mohly vzniknout po cestě mezi převodníkem a samotným MCU.

3.2. Návrh a řešení univerzální desky

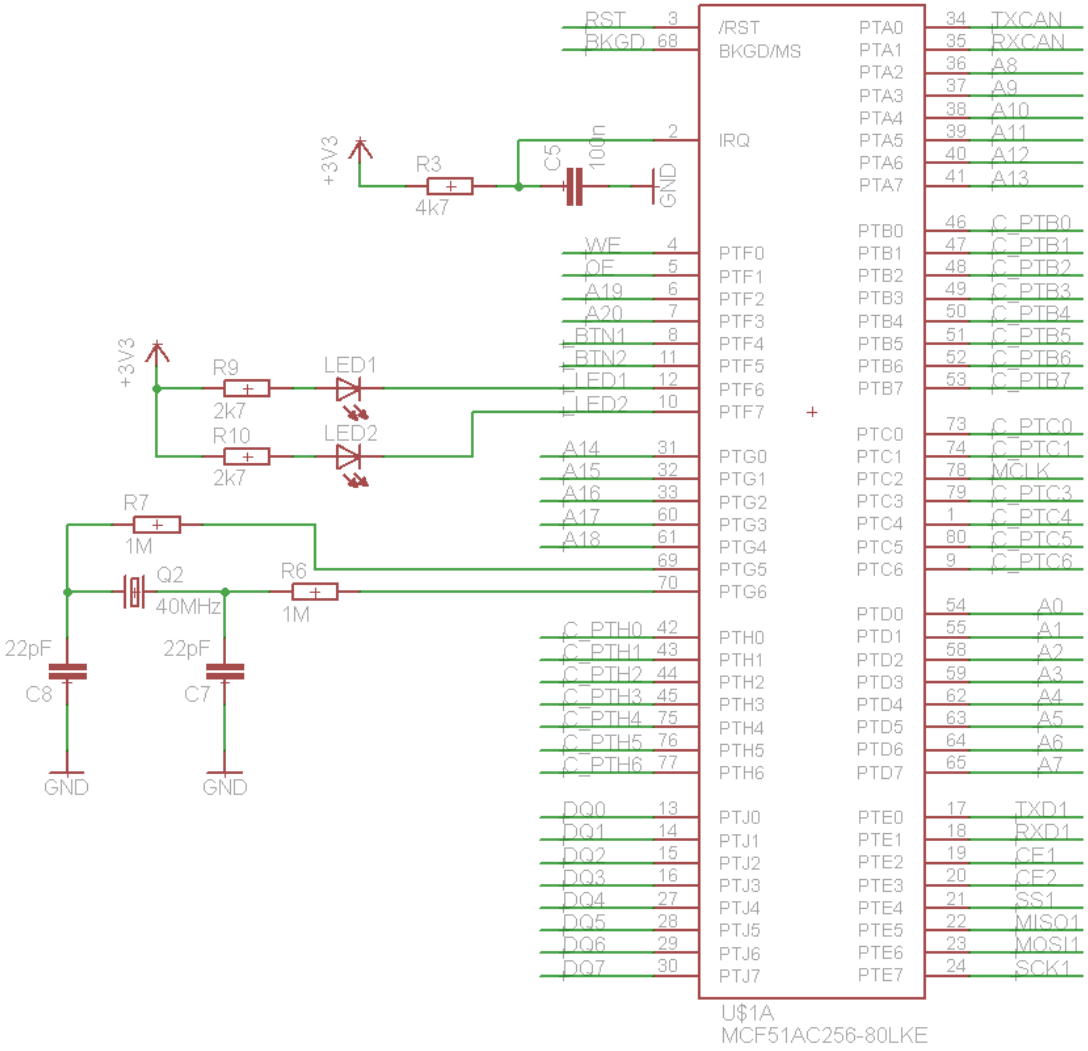
Univerzální deska s kontrolérem MCF51AC256 slouží jako mnohostranné zařízení pro různé aplikace. Je využívána k řízení polohovacích servomotorů, k analýze a zpracování obrazu a také k analýze a zpracování zvuku, což si klade za úkol tato diplomová práce. Vytvořené schéma zapojení je vypracováno v programu Eagle. Vodivé cesty desky jsou navrženy oboustranně a na každé z nich také ve více vrstvách. Toto řešení se užívá z toho důvodu, že je určen pro malé rozměry desky. Výsledná aplikace je zakomponována na modelu hlavy, kde se nachází celkem 3 univerzální desky plus další potřebné, jako jsou např. na ovládání pohonu, kamer, v případě této práce deska pro lepší zpracování zvukových signálů. Obrázek níže zobrazuje kompletaci obou desek pro správnou funkčnost v rámci zpracování zvuku. Menší zelená DPS je pomocí 34pinové svorkovnice připojena k hlavní desce s MCU. Tento konektor zároveň slouží jako mechanické upevnění pomocné DPS.



Obrázek 11 - Univerzální deska s deskou pro zpracování zvuku

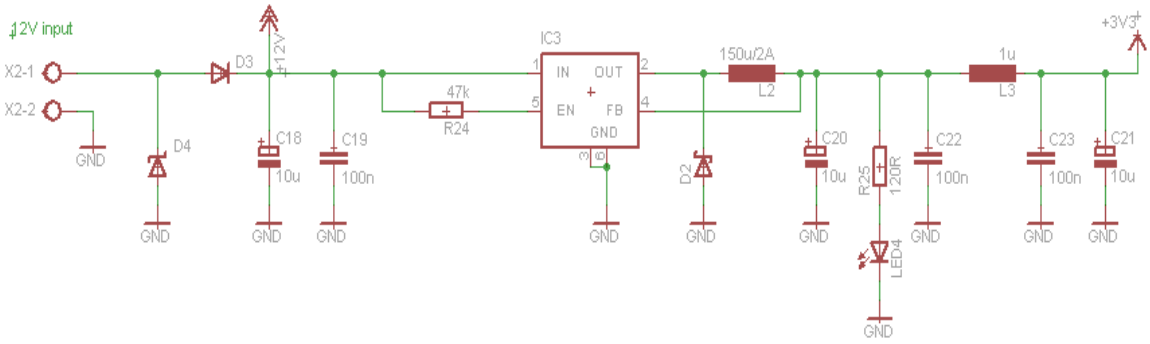
3.2.1. Schéma zapojení

V základním schématu je zapojen MCU s potřebnými obvodovými prvky pro správnou funkci. Zapojení napájecího napětí, správná stabilizace, zapojení krystalu pro nastavení kmitočtu procesoru atd. Zapojení procesoru vychází z doporučení od výrobce.[13] Ostatní vstupy a výstupy, které slouží k spojení do jednotlivých periferií MCU jsou vyvedeny k dalším obvodovým prvkům na DPS např. komunikace, slot paměťové karty. Kontakty, které nemají blíže specifikovanou funkci nebo jsou připravené pro připojení rozšíření, jsou vyvedeny na konektor společně s veškerým napájením.



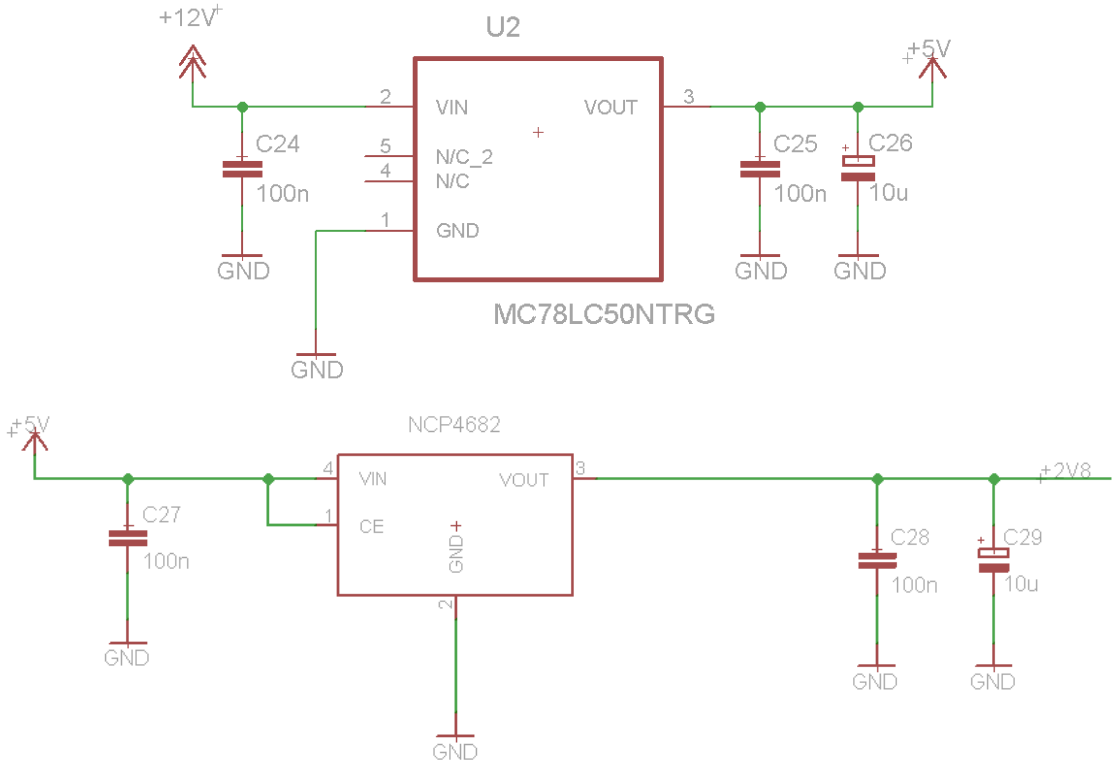
Obrázek 12 - Základní zapojení MCU

Část schématu ukazuje zapojení stabilizovaného zdroje, který napájí univerzální desku. Na vyvedený konektor pro připojení rozšiřujících desek jsou vyvedeny další úrovně napěťových hladin. Různé napěťové úrovně jsou vyvedeny pro větší univerzálnost desky a také pro to, že deska s kamerovým čipem potřebuje jiné napájecí napětí než deska pro zesílení zvuku.



Obrázek 13 - schéma zapojení zdroje s výstupem 3,3V

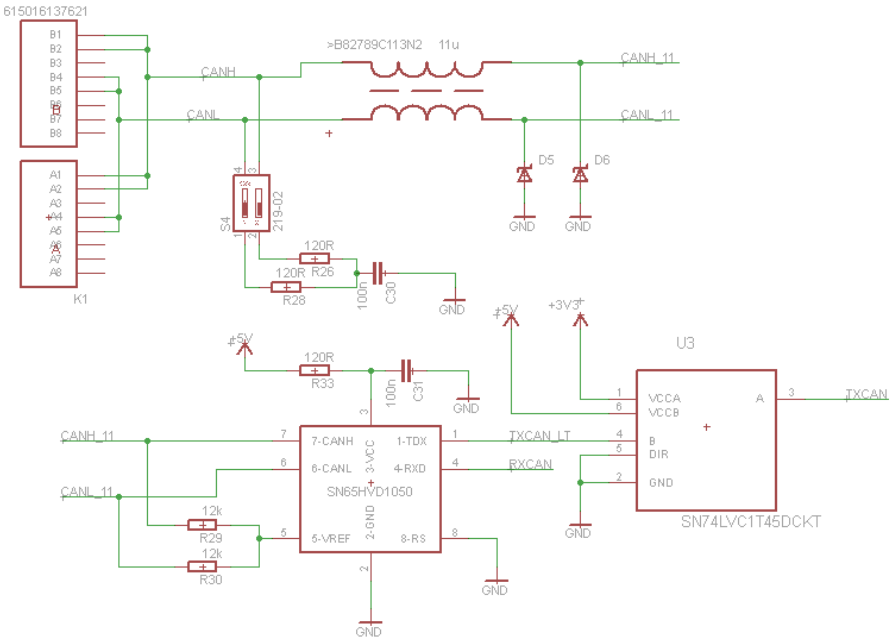
Napětí +12V je vyvedeno přímo ze vstupu ještě před snížením na 3,3V. Úroveň napětí +5V je získávána ze vstupního napětí snížením pomocí napěťového regulátoru MC78LC50NTRG. Napětí +2,8V se pomocí NCP4682 sníží z hodnoty +5V. Stabilizátor má podobné vlastnosti jako předešlý pro +5V. Hlavní výhodou obou prvků je to, že mají velmi nízký úbytek napětí při regulaci. Nedochází proto tak rychle k přehřátí součástky a jsou více odolné vůči poškození vyšším napětím.



Obrázek 14 - zapojení stabilizátorů na +5V a +2,8V

Použití komunikační sběrnice CAN

Řízení robotického zařízení vyžaduje více desek s MCU. Tyto desky mezi sebou komunikují pomocí sběrnice CAN. Hardwarové řešení je navržené pomocí dvou čipů SN65HVD1050. Tento převodník slouží pro předávání dat oběma směry, jak z procesoru do komunikační sítě, tak i pro data, která ze sítě přicházejí. Druhým převodníkem je součástka SN74LVC1T45DCKT, sloužící k převodu signálu posílaného z procesoru do komunikační sítě. Komunikační rychlost obou prvků se liší podle typu napájení. V našem případě využíváme při +3,3V komunikační rychlost až 210Mb/s na těchto čipech. V porovnání se sběrnicí CAN to je však zbytečné, protože komunikační rychlost má velikost 1Mb/s.[13]



Obrázek 15 - schéma zapojení CAN

CAN bus

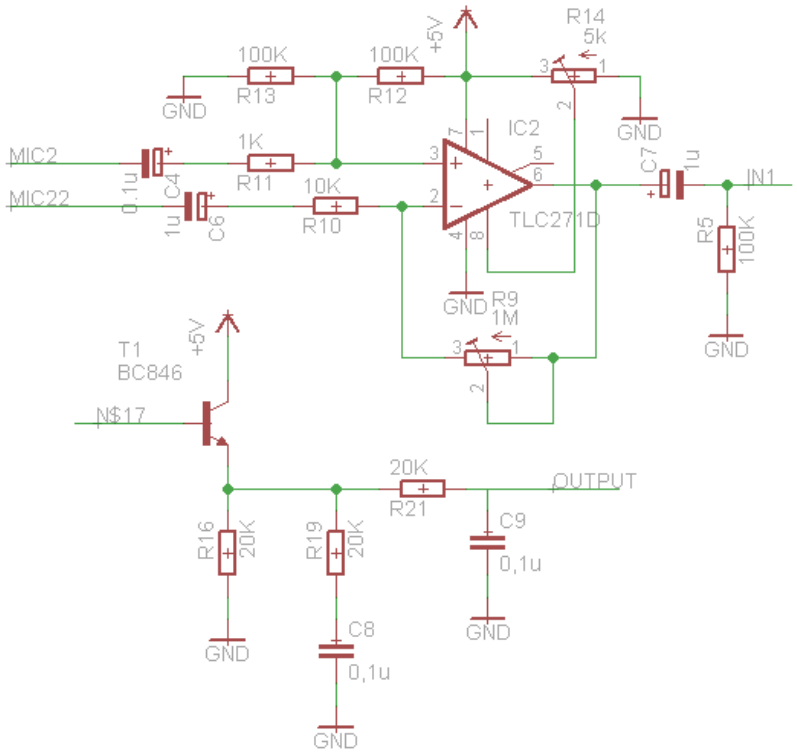
CAN (Controler Area Network) je sběrnice, využívaná nejčastěji pro vnitřní komunikační síť senzorů a ostatních funkčních prvků v automobilovém průmyslu. Z tohoto odvětví se rychle komunikace rozšířila do ostatní průmyslové automatizace. Jedná se o sériovou sběrnici vyvinutou společností Robert Bosh GmbH. Jako maximální komunikační rychlost přenosu po sběrnici se udává 1Mb/s. Elektrické parametry přenosu jsou určené normou ISO11898. Síťový protokol detekuje a opravuje přenosové chyby vzniklé okolním elektromagnetickým rušením. Potřebná data se posílají v rámcích, které můžou mít až 8 datových bajtů. Každý poslaný datový rámec obsahuje identifikátor, protože sběrnice CAN nemá žádnou adresu. Identifikátor určuje prioritu a obsah zprávy. Čím menší hodnotu má identifikátor, tím má zpráva vyšší prioritu. Díky této vlastnosti může zprávu číst několik zařízení zároveň. Když zpráva není určená pro nějaký typ zařízení, tak ji nečte a nechává ji pro ostatní zařízení jím určené. Metody „bitové arbitráže“, které se používají k identifikaci zpráv, jsou schopné analyzovat jakékoli problémy mezi stanicemi, co čekají na přenos.[14]

3.3. Návrh zesilovače vstupního signálu

Pro dosažení kvalitnějšího vstupního signálu z mikrofonů bylo použito zapojení se zesilujícími prvky. Zapojením mikrofonu na zesilovač TLC271, podle doporučeného zapojení přímo od výrobce, je dosaženo nejkvalitnějšího zesílení, aniž by byl signál zkreslen nebo ořezán v určitých frekvenčních mezích.[15] Vyrobený zesilovač je pomocí 34pinové patice připojen k univerzální desce. DPS se zesilovačem využívá z připojených pinů jen čtyři jako signálové propojení s procesorem na univerzální desce. Díky malým rozměrům DPS není potřeba ji mít uchycenou pomocí distančních sloupků. V tomto případě jako mechanické upnutí slouží konektor, kde je připojeno pět pinů jako signálové a napájecí. Ostatní piny slouží k propojení země s UD nebo nejsou zapojeny. V případě zapojení jen potřebných pinů by soudržnost u obou desek nebyla tak kvalitní a muselo by být použito jiné řešení uchycení.

3.3.1. Schéma zapojení zesilovače

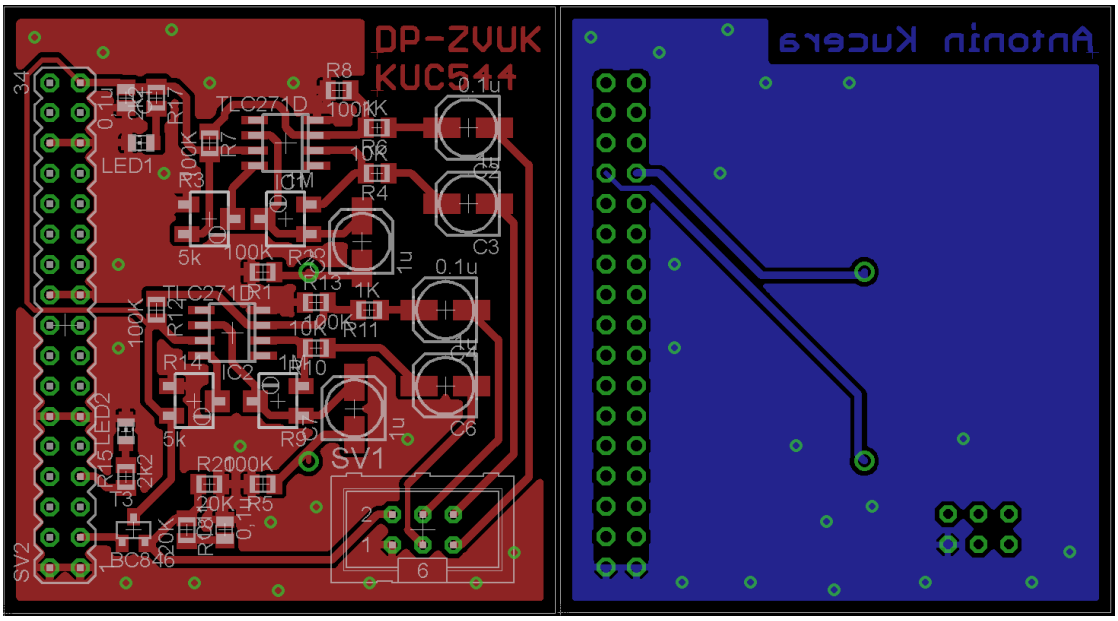
Pomocí *trimru R2* ve schématu měníme hodnotu zesílení. Použitím jednoduchých odporů by nebylo možné regulovat zesílení, díky *trimru* tak můžeme nastavit a doladit zesílení tak aby bylo co nejoptimálnější. Hodnota jednoho megaohmu je zvolena záměrně z důvodu dosažení vysokého zesílení. Zapojení elektrolytických kondenzátorů mezi mikrofon a vstup do zesilovače je ošetření rušení vzniklého na mikrofonu. Rušení typu hluku v prostoru je částečně eliminováno a díky zapojeným kondenzátorům jsou rušivé hodnoty vyhlazovány. Pomocí *trimru R3* je nastaveno referenční napětí pro správný chod zesilovače s označením TLC271D.



Obrázek 16 - Schéma zapojení zesilovače a zapojení výstupu na reproduktor

3.3.2. Návrh desky plošných spojů

Desky plošných spojů pro zesilovač je vytvořena v programu Eagle. Na obrázku níže v levé části (červeno-hnědá) jsou navržené cesty ze strany součástek (TOP). Pravá modrá část je ze strany spojů (BOTTOM). Deska je navržena oboustranně s vylitou zemí na obou stranách. Ta slouží k více věcem zároveň. Hlavní úlohou vylité země v nepoužitých částech desky je odrušení nežádoucích signálů vytvářených v okolí desky (z kabelů, vysokofrekvenčních součástek, atd.). Pomocí prokovených děr jsou tyto země spojené. Prokovené díry jsou na desce umístěny náhodně s jistým záměrem. Díky náhodnému uspořádání děr eliminujeme rušivé vířivé proudy. U takto malé aplikace by zřejmě nemusely vzniknout, ale u větších aplikací nebo při různých druzích signálů by nastal problém, a pak by se musel odstranit složitým způsobem, proto je lepší s tím počítat už při návrhu a výrobě.



Obrázek 17 - Návrh DPS vlevo TOP vpravo BOTTOM

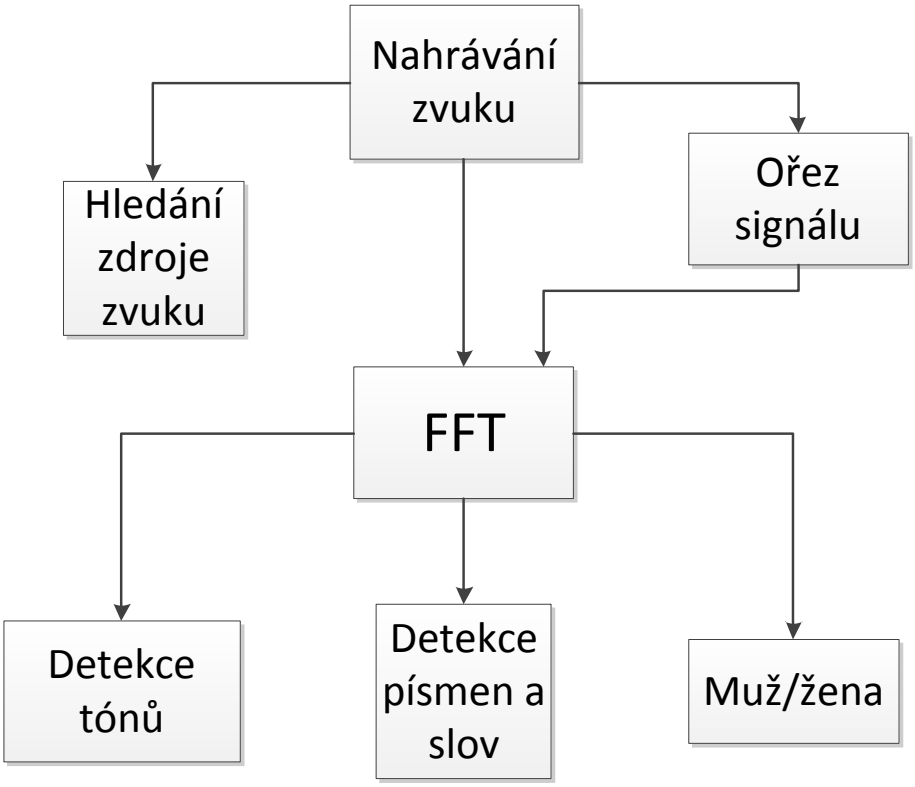
Velikost desky je eliminována na rozměry 47x53mm. Součástky na desce jsou umístěny s co možná nejmenšími vzdálenostmi mezi sebou, aby se zabránilo rušení, které může vzniknout v okolí umístěné desky. Kvůli získávání dat z mikrofonů, což jsou řádově milivolty, je odstranění rušení prvotní úlohou. Každý naindukovaný signál může rapidně ovlivnit výsledek měření, a tudíž by nemuselo docházet ke správné analýze dat. Původní testovací návrh byl vyroben pomocí osvitové metody v laboratorním prostředí školy. Návrh byl vytištěn na speciální folii, která se umístila na fotocitlivou desku. Po osvícení a vnoření do vývojové lázně byla přebytečná měď odleptaná v roztoku chloridu sodného. Prvotní návrh nebyl vyhovující z důvodu špatně rozmístěných součástek a také kvůli korozi mědi. Nebyla provedena žádná povrchová úprava po výrobě, tudíž se hůře pájely součástky a vlastnosti spojů nebyly ideální. Druhý návrh po opravení chyb se nechal vyrobit ve specializované firmě s povrchovými úpravami. Na části určené pro patice součástek byl nanesen cín, ostatní části byli zality to antioxidační vrstvy. Veškeré díry pro spoje, které nejsou pro SMD montáž, byly prokoveny.

4. Návrh a realizace metod pro analýzu zvukového signálu s detekcí mezí intenzity, hodnot frekvenčního pásma, detekcí jednoduchých povelů

Návrhem a realizací metod pro analýzu zvukového signálu je vytvořen program, který je formou jednoduchých metod a tříd vyvinut v prostředí Microsoft Visual Studio. Původní verze programu je tvořena pomocí objektově orientované platformy .NET neboli C# [16], ale kód je psaný tak, aby se s co nejmenšími úpravami dal přepsat do jazyka C [17]. Přepsání kódu do jazyka C a následného zkompilování a nahrání do MCU je provedeno v programu CodeWarrior od společnosti Freescale.[18]

4.1. Popis metod

Popisem metod je vyjádřeno, jak metody teoreticky fungují. Následné řešení programu ukazuje, jak jsou tyto metody podobné, nebo jaké jsou v nich rozdíly. Hledáním již použitých metod a doporučených od systémových zakladatelů lze jednoduše pochopit, jak tyto metody fungují a části z nich použít. Když jsou využívány knihovny pro zpracování operací, jsou napsané rozsáhlým kódem. Části kódů použitých v knihovnách jsou zároveň využitelné při psaní jednoduchých metod a v konečném řešení se odstraní potřeba využití složitých knihoven, které sice slouží pro mnohé využití, ale v jednoduchých aplikacích jsou zbytečné.



Obrázek 18 - Diagram propojení metod

4.1.1. Metoda nahrávání zvuku

Nahrávání zvuku můžeme provádět dvěma způsoby. Aplikováním předem nahraného souboru nebo použití dat v reálném čase přímo z mikrofonu. Použití uloženého souboru je dobré pro analyzování v programu pro PC z důvodu objemnosti a početnosti dat. Výhodnějším řešením je ukládání do pole po potřebné analýze, která funguje ve stejném čase, jako jsou ukládána data do pole s nepatrným zpožděním. Po naplnění pole se hodnoty analyzují a poté jsou znovu přepsány. Aby byla zaručena správná odezva při analýze, každý prvek pole má adresu a při čtení se mění hodnota adresy v poli. Po přečtení všech dat v poli je smyčka vyčítající data z pole vynulována a čte pole od začátku již s novými daty.

```
//pro nahravani
[DllImport("winmm.dll", EntryPoint = "mciSendStringA", CharSet =
      CharSet.Ansi, SetLastError = true, ExactSpelling = true)]
      private static extern int mciSendString(string lpstrCommand,
      string lpstrReturnString, int uReturnLength, int hwndCallback);
//pro prehravac
[DllImport("winmm.dll")]
     private static extern long mciSendString(string strCommand,
     StringBuilder strReturn, int iReturnLength, IntPtr hwndCallback);
```

V běžném použití programování aplikací pro PC je potřeba nejdříve naimportovat knihovny. Kódem uvedeným výše je přistupováno k systémovým knihovnám. V použití MCU techniky nám tato deklarace a přístupy odpadají. Díky programování pomocí programu CodeWarior s vestavěnými funkcemi definování vstupů a výstupů pomocí tzn. fazolí je mnoho funkcí automaticky vygenerováno. Díky vestavěné periferii ADC v MCU je velice jednoduché vyčítání převedených analogových dat. Pomocí následujícího kódu jsou vyčítány a následně ukládány data do pole, které je po naplnění daty dále analyzováno pomocí dalších metod.

```
for(b=0;b<257;b++)
{
  (void)AD1_Measure(TRUE);      // Konverze a čekání na výsledek
  (void)AD1_GetValue16(&value);       // uložení výsledku
   vOut[b]=(unsigned char) value;     //uložení hodnot do pole
}
```

Smyčka *for* je omezena na velikost 257 znaků z důvodu urychlení procesu analýzy. Při probíhajícím zpracování dat následujícími metodami toto množství stačí. Při každé opakující se smyčce procesoru jsou načtena nová data a analýza pokračuje od poslední hodnoty, tudíž při rychlosti procesoru je prodleva mezi čekáním na nová data minimální. Dalo by se říct, že zpracování dat probíhá téměř v reálném čase. Při použití ukládání většího objemu dat např. 1024 nebo 2048 je procesor zbytečně vytížen, co se týče dočasné paměti pro data a také vzniká moc velká prodleva mezi jednotlivými cykly. Analýza jednotlivých vzorků by byla jednotnější, ale časově náročnější. Z tohoto hlediska byl zvolena současná metoda ukládání a zpracování dat.

4.1.2. Metody rychlé Fourierovy transformace

Použitím normální Fourierovy transformace v programu by veškeré operace zabrali více času, a proto je používána rychlá Fourierova transformace (FFT). Metod psaní FFT existuje mnoho a většina jejich řešení se jeví jako zbytečně složitá.

```
while (n > mmax)
            {
                var wpr = Math.Cos(theta);
                var wpi = Math.Sin(theta);
                for (var m = 0; m < istep; m += 2)
                {
                    for (var k = m; k < 2 * n; k += 2 * istep)
                    {
                        var j = k + istep;
                        var tempr = wr * wav[j] - wi * wav[j + 1];
                        var tempi = wi * wav[j] + wr * wav[j + 1];
                        wav[j] = data[k] - tempr;
                        wav[j + 1] = data[k + 1] - tempi;
                        wav[k] = data[k] + tempr;
                        wav[k + 1] = data[k + 1] + tempi;
                    }
                    var t = wr; // trig recurrence
                    wr = wr * wpr - wi * wpi;
                    wi = wi * wpr + t * wpi;
                     }
```

Předchozí část kódu je stěžejním obecným tvarem jak naprogramovat FFT. Toto řešení není však ideální a přes svoji rychlou funkčnost je kód složitý. Použití u polí s velkým objemem dat se jeví jako nepraktické, protože cyklus trvá velmi dlouho. FFT se používá v aplikacích reálného času, kde hodnot v poli je málo a rychle se mění.

```
while (k < n)
{
        for (int i = 1; i <= n2; i++)
        {
              tr = xre[k + n2] * c + xim[k + n2] * s;
              ti = xim[k + n2] * c - xre[k + n2] * s;
              xre[k + n2] = xre[k] - tr;
              xim[k + n2] = xim[k] - ti;
              xre[k] += tr;
              xim[k] += ti;
              k++;
        }
        k += n2;
 }
```

Jak je vidět kód není o moc rozdílný. Základ je stejný. Pomocí smyčky for se přepisují data z pole, ve kterém jsou uloženy vzorky signálů. Každému vzorku se přiřazují hodnoty, které při vykreslení tvoří peaky, neboli jsou vnímány jako sloupce hodnot.

4.1.3. Metody hledání zdroje zvuku

Zdroj zvuku může mít různé umístění a díky zapojení dvou mikrofonů je možné detekovat umístění, jestli se objekt nebo osoba vydávající zvukový signál nachází na levé straně nebo pravé straně k pozici detekčního zařízení. Z každého mikrofonu se ukládají data do pole. Při porovnání polí hodnot se zjišťuje, v jakém se vyskytuje větší intenzita zvuku, tam je pak zřejmé, ve kterém směru se zdroj nachází.

```
for(i=0;i<257;i++)
{
      vypP=(unsigned char)vOutP[i+vypPi];
      vypPsouc = vypP+vypPsouc;
      vypPi++;
}
for(j=0;j<257;j++)
{
      vypL=(unsigned char)vOutL[j+vypLj];
      vypLsouc = vypL+vypLsouc;
      vypLj++;
}
if(vypLsouc>vypPsouc)
{
      Led6_PutVal(1);
      PutChar('1234'+(127-(vypLsouc-vypPsouc)));
}
if(vypLsouc<vypPsouc)
{
      Led7_PutVal(1);
      PutChar('1234'+(127+(vypPsouc-vypLsouc)));
}
```

Vyobrazený kód jednoduše popisuje, jakým způsobem se vyhodnocují veličiny pro detekci strany silnější intenzity zvuku. V první řadě musíme v kódu načíst data převedená z ADC převodníku. Tyto data jsou uložena do pole o velikosti 256 prvků. Po uložení těchto dat jednoduchým způsobem sečteme všechny hodnoty v poli. Tímto způsobem načteme a zpracujeme číselné hodnoty z obou kanálů ADC, tudíž tak dosáhneme výsledků z obou připojených mikrofonů. Dále jsou tyto hodnoty porovnány a v případě, že je některá z nich vyšší, tak se provede příslušná podmínka. Poměrově v závislosti na rozdílu hodnot se odešle hodnota, na kterou stranu má být hlava natočena. V reálném případě se bude hlava natáčet lineárně v závislosti a pohybujícím se zdroji zvuku číselná hodnota ´1234…´pro posílání pomocí UART je pro lepší rozpoznání, jaká zpětná vazba přišla do počítače. Při posílání více druhů dat tak jednoduše rozlišíme, že přišla hodnota souřadnic pro pozici hlavy.

4.1.4. Metody ořezu signálu

Každý zvukový signál má určitou intenzitu, také se skládá z vysokých a hlubokých tónů. Díky těmto vlastnostem můžeme ořezat hodnoty nad nebo pod určitou mez. V případě zapojení elektronických součástek odporů a kondenzátorů můžeme vytvořit dolnopropustný nebo hornopropustný filtr. Programové řešení je náročnější na výpočet skrze závislosti na převodu signálu pomocí FFT. Po převedení signálu pak už jen stačí ořezat nepotřebné frekvence.

4.1.5. Metoda detekce tónů stupnice C dur

Součástí práce je analyzování vstupních dat a v závislosti na vstupním signálu i reakce na detekované podněty. Detekce tónů stupnice C dur je mezistupeň k detekci řeči. V programu jsou uložena pole hodnot nahraných tónů stupnice. Příslušné hodnoty v polích jsou takové, které byly nejdříve převedeny z amplitudového spektra do frekvenčního pomocí FFT. Detekce jednotlivých tónů je provedena porovnáváním již uložených polí s reálně měnícím se polem hodnot. Když nastane shoda uloženého pole s reálným, je to vyhodnoceno jako detekování příslušného tónu. Shoda nelze určit na 100%, ale u pole (tónu), které má nejvíce shodných prvků je shoda brána jako maximální, a tím i detekován příslušný tón. Jako další součástí kontroly shody správného tónu využíváme detekci pomocí frekvence příslušného signálu podobně jako u akustiských ladiček různých hudebních nástrojů. Každý tón má danou svoji frekvenci, která je popsána v tabulce níže [19]

Tón	Pýthagorejci Hz]	Temperovaně [Hz]
C	262	262
D	295	294
E	332	330
F	349	349
G	393	392
A	442	440
H	497	494
C´	524	524

Tabulka 2 - Frekvence tónů stupnice C-DUR

Pomocí FFT, přes kterou převádíme měřený analogový signál, dokážeme určit danou frekvenci signálu. Při nalezení shodné frekvence při určování správnosti tónu si zvýšíme pravděpodobnost nalezení správného tónu analyzované stupnice.

```
if (((D1[x] - ZvukAnalyza.Form1.tolerance) <= _fft[x]) && ((D1[x] +
ZvukAnalyza.Form1.tolerance) >= _fft[x]))
{
      d++;
}
if (d > 0){
      abcD[l] = d;
      if (l == 4)
      {
            m = (abcD[0] + abcD[1] + abcD[2] + abcD[3] + abcD[4]) / 5;
            if (m >= 210)
            {
                  Console.Write("D1 ");
```

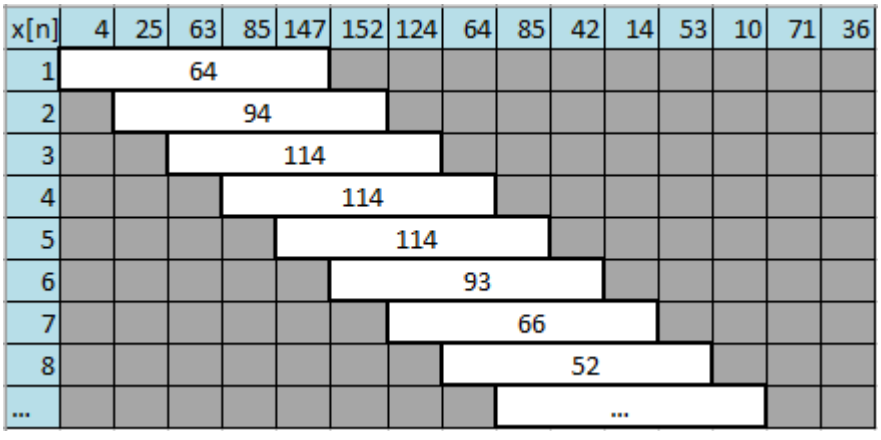
První částí kódu jsou porovnávána aktuální data s již uloženými. Při vyšší shodě je analyzováno pět aktuálních hodnot z nahrávaného a převedeného spektra pomocí FFT. Tyto hodnoty jsou sečteny a pokud je průměrná výsledná hodnota vyšší než nastavená mez, je signál vyhodnocen jako určitý tón stupnice. Tento způsob má minimálně 80% úpěšnost.

4.1.6. Metoda hledání maxima (tlesknutí)

Algoritmus pro hledání mezních hodnot v první řadě vyhledává maximální a minimální hodnoty v jednotlivých polích. Nalezení maxima nebo minima je důležité při nacházení řečových signálů, u kterých jsou tyto hodnoty podstatné pro jednotlivá písmena či jejich spojení ve slova. Každé vyřčené písmeno jakékoli abecedy má určitou intenzitu, co se týče jeho hlasitosti vyřčení. Různou výšku intonace a různou mezní hodnotu při dané frekvenci. Poměrem maxim a minim při daných frekvencích lze zjistit, o jakou hlásku se jedná, a jestli to je jen hláska nebo spojení několika hlásek dohromady. Dalším hledaným aspektem mezních hodnot je počátek a konec vyřčených písmen ve slovech. Při rozpoznávání slov je nutné nejdříve slovo rozčlenit na dílčí části (písmena) a z těch lze následně analyzovat dané slovo. Analýza celých slov je také možná, ale je potřeba velká databáze slov a díky tomuto problému je snadnější analýza pomocí písmen. Zde stačí databáze podstatně menší, pouze na počet písmen abecedy v daném jazyce, kde se mnohá písmena prolínají abecedami několika jazyků zároveň. Tím při tvorbě databází pro více národností klesá náročnost vytváření a eliminuje se jejich velikost.

```
if (aktivni == true)     //porovnani pulu za sebou
{
        c++;
        if (_wave[h - 1] >= _wave[h]){
              UdalostAktiv=true;
        }
        if (c == 5){
              aktivni = false;
              UdalostAktiv = false;
        }
 }
 h++;
```

Následující obrázek vysvětluje posloupnost analýzy hledání maxima, a tudíž i zajištění přesnější analýzy. Tento princip je možné použít i pro analyzování mluveného slova při hledání začátků a konců písmen. Bílá oblast označuje 5 aktuálních analyzovaných prvků v daném cyklu. Hodnota ve středu zobrazuje vypočítaný průměr. Když v určeném počtu cyklů odpovídá průměrná hodnota vyšší než stanovená mez, je signál vyhodnocen jako nalezení maxima.



Obrázek 19 - Princip hledání maxima z hodnot v poli

4.1.7. Metoda detekce mluvených písmen a slov

Implementovány byly dva způsoby jak detekovat a následně použít mluvené slovo. Prvním řešením bylo využití knihovny *speech.lib*, která je zakomponována přímo do operačního systému Windows. Jako druhé a finální řešení byl implementován zdrojový kód od společnosti Google a.s., který je volně dostupný k využívání.

Detekce pomocí Windows knihovny

Nevýhodou tohoto řešení je pomalá analýza nahrávek a také její závislost čistě na anglickém jazyce. Testovací program byl vytvořen v konzolové aplikaci pro rychlejší zpracování a ověření, že tento způsob není adekvátní pro použití v synchronizaci s procesorem na vytvořeném hardwaru.

```
static SpeechRecognitionEngine _recognizer = null;

static void RecDoTextu()
{
       _recognizer = new SpeechRecognitionEngine();
       _recognizer.RequestRecognizerUpdate();
       _recognizer.LoadGrammar(new Grammar(new GrammarBuilder("exit")));
       _recognizer.RequestRecognizerUpdate();
       _recognizer.LoadGrammar(new DictationGrammar());
       _recognizer.SpeechRecognized+=srectotext;
       _recognizer.SetInputToDefaultAudioDevice();
       _recognizer.RecognizeAsync(RecognizeMode.Multiple);
}
static void srectotext(object sender, SpeechRecognizedEventArgs e)
{
       if (e.Result.Text == "exit")
       {
              manualResetEvent.Set();
              return;
       }
       Console.WriteLine("Tvoje slovo: " + e.Result.Text);
}
```

Vytvořená statická proměnná *_recognizer* slouží jako delegát pro napojení metod obsažených v systémové knihovně, která slouží k rozpoznávání mluvené řeči a následně překládá na text. Prvním řádkem v metodě otevřeme přístup ke třídě *SpeechRecognitionEngine,* dále zavoláme metodu pro nahrávání, neboli spustíme obnovování dat při nahrávání. Další částí načteme novou třídu *DictationGramar* s vlastnostmi poděděnými z knihovny od stejně pojmenované třídy. Pomocí volání *SpeechRecognized* uložíme převedenou řeč na text do metody pro zobrazení výsledku. Vyvoláním metody *SetInputToDefaultAudioDevice()* je nastaven zdroj zvuku, z něhož má systémová knihovna načítat data na defaultně definovaný mikrofon. Záleží však na nastavení operačního systému v PC. Pokud má být použit interní mikrofon integrovaný do HW počítače (v případě, že jej má) nebo připojený do externího vstupu. Metoda vypisující analyzovaný text do konzole obsahuje ještě navíc část kódu pro ověření, jestli je ještě detekována řeč (jakékoli zvýšené hodnoty oproti průměrné hodnotě před detekcí). Když se hodnota sníží na průměr (šum okolí), aplikace vrátí textovou hodnotu.

Detekce pomocí Google

Aplikace je díky tomuto řešení závislá na připojení k internetu, ale když pomineme tento nedostatek, získáváme multifunkční zařízení na všechny dostupné jazyky světa, analýza řeči a následná zpětná vazba se zkrátila z několika desítek sekund na sekundy. Principiálně nahrávaná data ukládáme do souboru typu flac nebo wav, který je zasílán na předem definovanou url adresu. Po odeslání souboru v několika sekundách přijdou zanalyzovaná data formou textu, který dále analyzujeme pro účely zpětné vazby. Systém je dále vybaven modularitou pro urychlení zpracování dat a budoucí odstranění nebo omezení závislosti na internetovém analyzování. Každá nahrávka je dočasně uložena do paměti. V tabulce níže jsou zobrazeny dostupné jazyky pro potřebnou analýzu.[19]

jazyk	syntaxe	jazyk	syntaxe
Afrikaans	af	Irish	ga
Albanian	sq	Italian	it
Arabic	ar	Japanese	ja
Azerbaijani	az	Kannada	kn
Basque	eu	Korean	ko
Bengali	bn	Latin	la
Belarusian	be	Latvian	lv
Bulgarian	bg	Lithuanian	lt
Catalan	ca	Macedonian	mk
Chinese Simplified	zh-CN	Malay	ms
Chinese Traditional	zh-TW	Maltese	mt
Croatian	hr	Norwegian	no
Czech	cs	Persian	fa
Danish	da	Polish	pl
Dutch	nl	Portuguese	pt
English	en	Romanian	ro
Esperanto	eo	Russian	ru
Estonian	et	Serbian	sr
Filipino	tl	Slovak	sk
Finnish	fi	Slovenian	sl
French	fr	Spanish	es
Galician	gl	Swahili	sw
Georgian	ka	Swedish	sv
German	de	Tamil	ta
Greek	el	Telugu	te
Gujarati	gu	Thai	th
Haitian Creole	ht	Turkish	tr
Hebrew	iw	Ukrainian	uk
Hindi	hi	Urdu	ur
Hungarian	hu	Vietnamese	vi
Icelandic	is	Welsh	cy
Indonesian	id	Yiddish	yi

Tabulka 3 - Tabulka použitelných jazyků pro přepis řeči do textu

Možnosti použití Google API

Google API je nástavba od společnosti Google. Tato komponenta slouží jako rozšíření webových klientů a www stránek, které ji mohou bezplatně využívat. Její hlavní funkcí je snímat hodnoty z mikrofonu, analyzovat v daném jazyce a vrátit hodnotu v podobě textového řetězce. Nahraná data se nejdříve musí uložit do datového typu *flac*, který bývá použit u audionahrávek na CD nosičích, nebo také do typu *wav*, který je jednoduchým audio datovým typem. Před uložení dat do daného datového typu, musí být nadefinovány vzorkovací frekvence, a také kolik musí být bitů v jednom vzorku. V tabulce níže jsou zobrazeny hodnoty vzorkovaní, zvýrazněné hodnoty jsou nejčastěji používané.

vzorek/sec	bit/vzorek
8000	1024
11025 (telefonní kvalita)	2048
22050 (rádiová kvalita)	4096
32000	**8192**
44056	**16384**
44100 (CD kvalita)	32768

Tabulka 4 - Vzorkovací frekvence nahrávaného signálu

Po nadefinování vzorkovací frekvence a bitové délky vzorků jsou nahrávána data z mikrofonu do *bufferu* s nastaveným datovým tokem. Toto pole typu *wav* je konvertováno do datového typu *flac*, jak je zobrazeno v následujícím kódu.

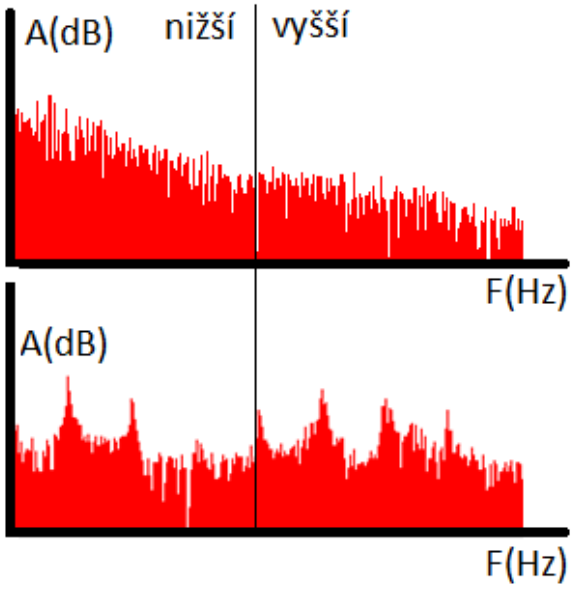
```
int sampleRate;
var flacName = Path.ChangeExtension(WavName, "flac");
FlakeWriter audioDest = null;
IAudioSource audioSource = null;
try
{
        audioSource = new WAVReader(WavName, null);
        AudioBuffer buff = new AudioBuffer(audioSource, 0x10000);
        audioDest = new FlakeWriter(flacName, audioSource.PCM);
        sampleRate = audioSource.PCM.SampleRate;
        while (audioSource.Read(buff, -1) != 0)
        {
              audioDest.Write(buff);
        }
 }
 finally
 {
        if (audioDest != null) audioDest.Close();
        if (audioSource != null) audioSource.Close();
 }
        return sampleRate;
```

Po konverzi do datového typu se správnou vzorkovací frekvencí jsou data navrácena a poslána na následující odkaz.

```
<https://www.google.com/speech-
api/v1/recognize?xjerr=1&client=chromium&lang=cs>
```

4.1.8. Metoda rozhodování muž/žena

Testování rozlišení mužského a ženského hlasu funguje následujícím způsobem. Po nahrání zvukových dat a uložení do pole se pole hodnot, prezentující amplitudové spektrum, převede pomocí rychlé Fourierovy transformace, kde vznikne další pole hodnot prezentující frekvenční spektrum. Pomocí určení vyšších a nižších frekvencí rozhodujeme, jedná-li se o ženský nebo mužský hlas viz obrázek níže.



Obrázek 20 - Frekvenční spektrum

Horní graf prezentuje hlubší hlas a spodní vyšší. Podle dělicí čáry určujeme, jedná-li se o ženský nebo mužský hlas. Vycházíme z faktu, že ženský obsahuje vyšší frekvence a mužský nižší. Je-li ve vysokých frekvencích obsaženo větší procento než na straně s nižších frekvencí, systém vyhodnotí tento signál jako ženský hlas. Nejrychlejší určení poměru obsažených frekvencí je pomocí funkce zprůměrování hodnot pro nižší a vyšší frekvence.

Algoritmus rozdělí výsledné pole na dvě části od stanovené meze, což je zhruba polovina. Zanalyzuje nejdříve první část dat, která sečte a uloží do mezi paměti. Následně zpracuje druhou část dat, se kterými provede stejnou operaci. Po té porovná obě výsledné hodnoty a poměrově vyšší hodnota, určí s přibližnou pravděpodobností jakého pohlaví je mluvící osoba. Následujícím kódem je pospaná zmíněná metoda, která je podobná hledání zdroje zvuku.

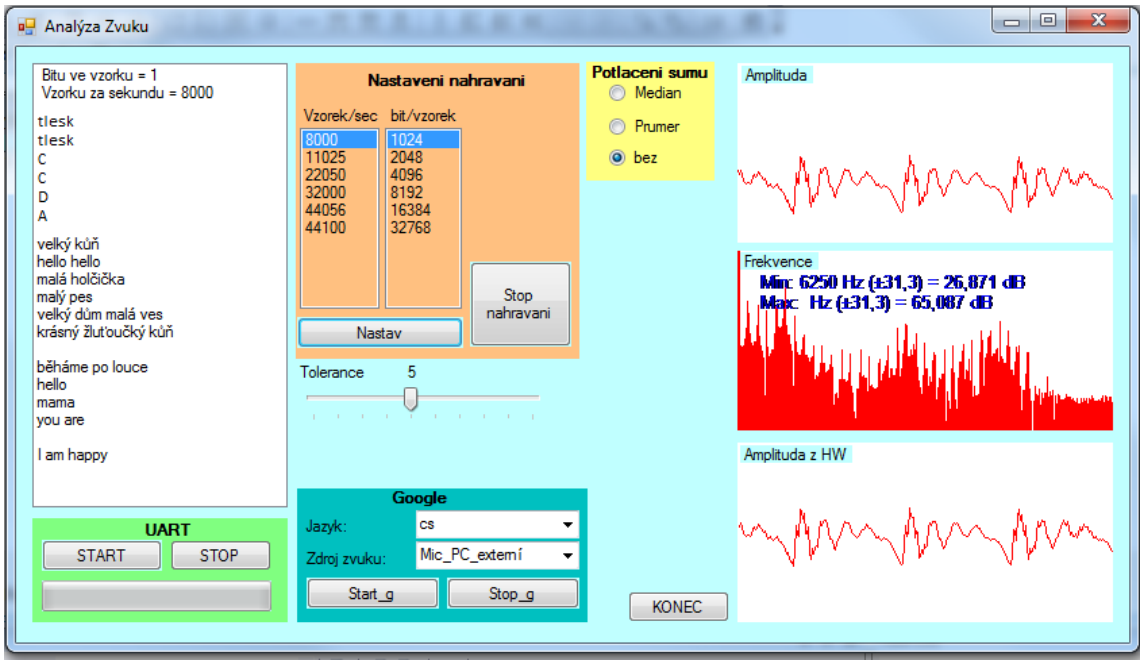
```
for(y=0;y<512;y++){
      if(y<256){
      rozV1=(unsigned char)vRozV1[y+rozV1q];
      rozV1souc = rozV1+ rozV1souc;
      rozV1q++;
}
if(rozV1souc > rozV2souc){
      PutChar('5678'+(256-( rozV1souc - rozV2souc)));
}
```

4.2. Implementace programu pro prostředí Windows

Vytvořené vizuální okno pro analýzu zvuku by formováno pomocí inteligentního nástroje zvaný GUI. Usnadňuje sledování analýzy zvuku a správný odečet hodnot, které jsou potřeba měřit.

4.2.1. Popis aplikace

V levé horní části okna se nám do konzolového okna vypisují požadavky nebo odezvy aplikace jako zpětná vazba povelů. Pomocí vizuálního prvku *listbox,* můžeme vybrat hodnotu, kolik vzorků za sekundu a bitů ve vzorku má být ukládáno při nahrávání do *bytového* pole, ze kterého jsou čtena data a následně analyzována. V horní střední části se nachází panel „Potlačení šumu“. Zde jsou na výběr dvě možnosti jak eliminovat šum v signálu. První možností pomocí funkce je median, která omezuje šum signál na střední hodnotu. Druhá funkce pomocí průměru funguje na celé pole zároveň a principem součtu a zprůměrování hodnot potlačí rovněž šumové složky. Obě metody ovlivní výstup signálu, ale hledané užitné spektrum není ovlivněno natolik, aby se nedalo s těmito hodnotami pracovat. Grafy v pravé části okna zobrazují amplitudové, frekvenční spektrum nahrávaných dat z mikrofonu počítače a amplitudové spektrum nahrávaného zvuku z HW části. Tyto data jsou do PC posílány pomocí UART. Frekvenční spektrum se mění v závislosti amplitudového signálu, ze kterého je frekvenční signál převáděn.



Obrázek 21 - Vzhled aplikace v programu Visual Studio

Panel v levé spodní části slouží ke spuštění a zastavení příjmu dat z komunikace UART. A poslední panel s označení Google slouží k nastavení analyzovaného jazyka od mluvčího. Dále k nastavení, od kterého mikrofonu mají být data přijímány a dále zasílány k analýze. Tlačítky „Start_g“ a „Stop_g“ spouštíme nahrávání a komunikaci se serverem společnosti Google.

4.2.2. Nahrávání a analýza zvukové signálu

Prvotní řešení nahrávání zvukového signálu bylo tvořeno pomocí metod v programu C#. Nahrávání bylo provedeno dvěma způsoby. Jeden otevření a nahrání již existujícího zvukového souboru. Druhým způsobem byla metoda, která funguje téměř v reálném čase. Základním principem ukládání zvukového signálu, po převedení z analogového spektra do digitálního, je uložení do pole typu *byte[]*. Toto pole se plní vzorky signálu s hodnotami 0 až 65535.

Ukládání do pole typu byte

Ukládání do pole je řešeno v reálném čase, tedy při připojeném zařízení ukládá hodnoty do stanoveného *bytového* pole. Velikost *bytového* pole je pevně nastavena na velikost 512 prvků. Po uložení převedených hodnot do pole jsou dále tyto data zpracovávána pro lepší analýzu dat. Zdání ukládání dat do pole je takové, že se ukládají data od nultého prvku pole po prvek maximální velikosti pole. Zároveň jsou tyto data zpracovávána a následně analyzována. Například když se uloží první prvek pole tak to vypadá, že je zrovna analyzován, vykreslen do grafu nebo převeden Fourierovou transformací. Skutečnost je taková, že pole je nejdříve naplněno prvky vystupujícími z AD převodníku. Až je *bytové* pole naplněno 512 prvky, je předáno dalším procesům, které analyzují nebo zpracovávají uložená data, což je popsáno v použitém kódu.

```
byte[] recBuffer = new byte[size];
System.Runtime.InteropServices.Marshal.Copy(data, recBuffer, 0, size);
_streamMemory.Write(recBuffer, 0, recBuffer.Length);
if (_recorderBuffer == null || _recorderBuffer.Length != size)
      _recorderBuffer = new byte[size];
if (_recorderBuffer != null)
{
      System.Runtime.InteropServices.Marshal.Copy(data,
      _recorderBuffer, 0, size);
      if (_isPlayer == true)
            _streamOut.Write(_recorderBuffer, 0,
            _recorderBuffer.Length);
            audio.Process(ref _recorderBuffer);
      ...
     audio.Grafamlitudy(ref pictureBoxAmp);
     audio.GrafFFT(ref pictureBoxFFT, 8000);
```

První částí je vytvořeno pole typu *byte* na zadanou velikost. S tímto polem dále pracujeme jako s *bufferem* (dočasné ukládání nahraných dat). Dále předáme hodnoty systémové knihovně pro správné předávání hodnot. Pomocí uložení hodnot do proměnné *_streamMemory* předáme již uložená data z mikrofonu zároveň s hodnotou velikosti *bufferu*. Pokud má proměnná *_recorderBuffer* nulovou hodnotu (není vytvořen) nebo se hodnota velikosti bufferu nerovná s požadovanou, je proměnné znovu vytvořena velikost na požadovanou. Pokud je proměnná bufferu aktuální, jsou do ní ukládána data načtená z mikrofonu přes pomocné proměnné. Po načtení všech dat podle velikosti pole jsou tyto hodnoty předány metodám a třídám pro analýzu zvuku, nejprve pro grafy, poté pro FFT a nahrávání pro analýzu řeči přez Google.

4.2.3. Analýza uloženého zvukového souboru

Metoda načtení již stávajícího souboru se zvukovým signálem, se stává jednodušší v případě použití PC. Tento soubor lze pomocí dialogového okna otevřít a nahrát do námi určené proměnné. Následující kód znázorňuje, jakým příkazem je možno otevřít požadovaný zvukový soubor. První řádek otevře dialogové okno. Druhý řádek je filtr, který zaručuje to, že při výběru vidíme jen soubory s koncovkou wav. Např. Chceme-li otevřít zvukový soubor s názvem „ukázka1“ a v PC jsou uloženy dva soubory s tímto názvem, jeden má koncovku mp3 a druhý wav, při výběru se nám díky filtru zobrazí jen soubor „ukázka1.wav“.

```
OpenFileDialog a = new OpenFileDialog();
a.Filter = "wave files|*.wav";  //filtr pro otevření jen wav souboru
a.Title = "Select a wave File";    //Nápis nahoře v dialogovém okně
```

Podmínkou, která je vidět níže, ověřujeme, jestli byl soubor nahrán nebo ne. Tím se vyvarujeme kolapsu programu při nesprávném nahrání souboru. Pokud by měl být nahrán nesprávný soubor nebo soubor se špatným formátem dat díky této podmínce se operace neprovede.

```
if (a.ShowDialog() == DialogResult.OK)
```

Po správném nahrání souboru a uložení do proměnné, zavoláme metodu převádějící zvukový soubor do již zmíněného pole typu *byte*.[20] Následujícím příkazem určíme cestu, která nám říká, ze které proměnné se má nahrát zvukový soubor do pole.

```
    wav = File.ReadAllBytes(NameSound);
```

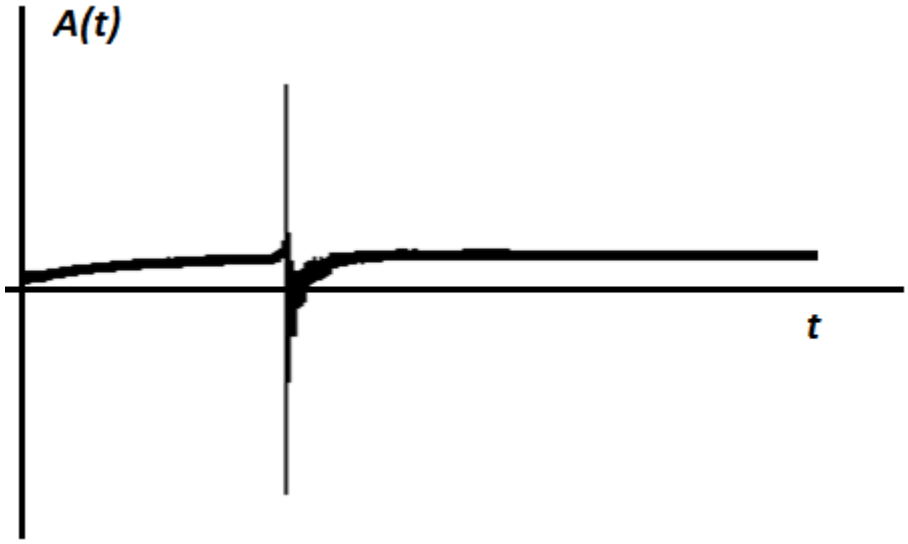
Pomocí smyčky „while“ postupně uložíme jednotlivé vzorky zvukového signálu do námi vytvořeného pole. Příkaz znamená, že se mají postupně nahrávat jednotlivé vzorky signálu, navíc specifikuje velikost pole tak, aby se chovalo jako dynamické. Díky nastavení automatického zvětšení může pole pojmout různé počty dat, aniž by program zkolaboval tím, že by byla překročena maximální hodnota pole.

```
while (!(wav[iPos]==100&& wav[iPos + 1]==97 && wav[iPos + 2]==116))
  {
     iPos += 4;
     int iChunkSize=wav[iPos] +wav[iPos+1] *256 +wav[iPos + 2] *65536;
     iPos += 4 + iChunkSize;
     wwfft = wav[iPos];
  }
```

Celá délka načteného pole je nakonec předávána pro další zpracování. Díky datům, která jsou statická a nemusíme řešit načítání nových dat, není třeba eliminovat dobu zpracování. Pomocí FFT, která přijala data analogového spektra, jsou hodnoty zpracovány a v druhém grafu zobrazeny jako spektrum frekvenční.

Hledání maxima

Metodou hledání maxima v poli se snažíme analyzovat výkyvy ve zvukovém signálu. Tato analýza v prvotní fázi sloužila jako detekce tleskání. Analýza probíhala v dlouhodobém porovnávání zvukového signálu u statického pole. V poli byl nahrán zvukový soubor, ve kterém byl jeden pulz označující „tlesk“ viz obrázek níže.



Obrázek 22 - Zobrazení tlesknutí v grafu

Níže uvedeným kódem je analyzováno zadané pole nebo jím také jde analyzovat nahrávaný zvuk přímo z mikrofonu. První částí metody určíme zdrojový soubor, odkud mají být použity hodnoty. Nastavíme *buffer* pro dočasné uležení dat. Deklarací proměnných si určíme hodnoty maxima a hodnoty pro pozdější spuštění a ukončení analýzy zvukového signálu.

```
      // nastavení bufferu
int blockLength = (int)Bass.BASS_ChannelSeconds2Bytes(decodingStream,
1f);
float[] buffer = new float[i / 4]; // 32-bit!
float mypeak = 0f;
float maxPeak = 0f;
long start = 0L;
long end = long.MaxValue;
```

Tento uvedený typ kódu není zcela ideální jako metoda hledání maxima popsaná výše v dokumentu. Tato metoda je sice funkční, ale použitelná pouze pro zjištění maxima. Při opakovaném signálu, jako je mluvená řeč, kdy během několika desítek vzorků je maxim více, tak metoda analyzuje nejvyšší hodnotu. V důsledku toho vrátí hodnotu, že nastalo tlesknutí, ale přitom spektrum přijímaných dat vykazuje hodnoty pro jiný typ analýzy. Pro použití datových souborů je tato metoda ideální. Pro analýzu v reálném čase je lepší použití novější metody, než je tato, kde se nastavuje i šířka pulzu, tudíž omezujeme vzik chyby nesprávné analýzy na minimum. Pokud je nadefinována šířka pro analýzu hledání maxima shodná s přicházejícím signálem a mezní hodnoty jsou dostatečně vysoké potom program vyhodnotí signál jako „tlesknutí“.

Hledání maxima ve statických datech

Smyčkou *while* postupně čteme data z pole a porovnáváme jejich hodnoty. Při podobných datech, které spíše signalizují šum než potřebný signál, jsou hodnoty ignorovány. Při větším rozdílu mezi hodnotami porovnává, jaké to jsou hodnoty, a když dosáhne podmínky, že maximální hodnota je stejná jako námi hledaná, tak pošle informaci o nalezení maxima. Tato informace je brána jako výkyv ve zvukovém signálu a předpokládáme, že je to hledané „tlesknutí“. Pomocí zobrazeného kódu níže

```
while (start < end)
{
      // skenování zvuku v poli (vrácení PCM dat z bufferu)
      int len = Bass.BASS_ChannelGetData(decodingStream, buffer,
      blockLength | (int)BASSData.BASS_DATA_FLOAT);

      if (len < 0)
            start = end;
      else
      {
            start += len;
            len /= 4;
            // hledání maxima ve zvukovém souboru

            for (int a = 0; a < len; a++)
            {
                  mypeak = Math.Abs(buffer[a]);
                  if (mypeak > maxPeak)
                  maxPeak = mypeak;
            }
      }
}
```

Tento způsob hledání „tlesknutí“ je však nevyhovující, protože část kódu je objektově orientovaná, a tudíž v potřebném kódu jazyka C nepoužitelná. Navíc řešení pomocí již nahraného zvukového souboru a jejího nahrávání a analyzování je nepraktické pro užití v MC. Předem nahraný soubor, i přes nízkou vzorkovací frekvenci, má v sobě uloženo příliš mnoho vzorků a jejich zpracování je velmi náročné. Použití MC, které má podstatně menší výpočetní výkon, nepřipadá v úvahu. Proto je řešení nahrávání zvuku v reálném čase výhodnější a pro použití analýzy v MCU použitelné.

Analýza řeči tímto způsoben nebyla řešena z důvodu nepraktičnosti. Principiálně by to šlo, protože rozdíl mezi nahrávanými daty v reálném čase nebo již vyčtenými z nahraného souboru žádný není. U obou typů musíme nastavit vzorkovací frekvenci bitovou délku vzorků. Rozdíl mezi nimi je, že statickou nahrávku musíme nejdříve pomocí jakéhokoli programu zaznamenat, poté jej musíme otevřít, vyčíst data a ty pak teprve můžeme analyzovat. U nahrávání v reálném čase nám tento složitý proces odpadá.

4.2.4. Analýza zvuku v reálném čase

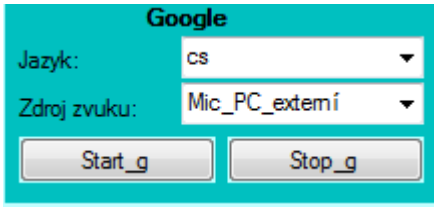
Nahrávání zvukového signálu přímo z mikrofonu a ukládání do pole v reálném čase má u MCU častější využití než nahrávání do souboru a jeho následné analyzování v průběhu času. V použití reálného času je možno hodnoty hned číst a již nepotřebné přečtené a zanalyzované hodnoty přepsat novými. Díky této funkci nahrávaná data spotřebují podstatě menší část paměti v úložném prostoru využívaným procesorem. Aplikace popsaná výše v metodách a grafickém návrhu aktuálně funguje na principu reálného času dvěma způsoby. Program pro použití v operačním systému Windows zároveň slouží k propojení HW části s internetem, a také jako vizuální prvek pro zobrazování zpětných vazeb od HW.

Popis částí programu pro analýzu pomocí PC

Hlavní součástí programu pro analýzu v PC je panel pro nastavení vzorkovacích frekvencí. Po výběru hodnot z obou *listboxů* můžeme kliknout na tlačítko „Nastav“. Po stisknutí tlačítka spustíme nahrávání dat z mikrofonu.

Obrázek 23 – Panel nastavení pro nahrávání

Tato data jsou předávána grafům pro zobrazení amplitudového spektra dále FFT, která převede amplitudové spektrum na frekvenční pro další graf a probíhající nastavené detekce. Při tomto procesu jsou detekovány klavírní tóny stupnice C dur, muž/žena a jsou detekovány výkyvy v signálu v podobě „tlesknutí“ nebo jakýchkoli krátkých, ale výrazných zvuků.

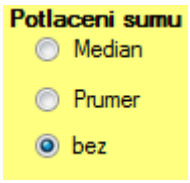


Obrázek 24 – Panel spuštění analýzy pomocí Google

Po stisknutí tlačítka na panelu „Google“ se probíhající lokální analýza zastaví a nahrávaná data jsou předávána pro analyzování řeči na server od společnosti google. Tato data jsou do doby, než stiskneme tlačítko „Stop_g“, které obnoví zpět lokální analýzu zvuků a tónů. Data z mikrofonu jsou nahrávána a po zjištění, že osoba přestala mluvit data se zašlou pro analýzu. V několika okamžicích přijde zpětná vazba v podobě textového řetězce.

Panelem pro potlačení šumu můžeme dvěma metodami šum v signálu omezit. Zde jsou na výběr dvě možnosti, jak eliminovat šum v signálu. První možností pomocí funkce medián, která omezuje šum signálu na střední hodnotu, jen u dvou prvků po sobě jdoucích. Druhá funkce pomocí průměru pracuje na celém poli zároveň a principem součtu a zprůměrování hodnot potlačí taky šumové složky. Obě metody ovlivní výstup signálu, ale hledané užitné spektrum není ovlivněno natolik, aby se nedalo s těmito hodnotami pracovat.

.



Obrázek 25 - Panel potlačení šumu

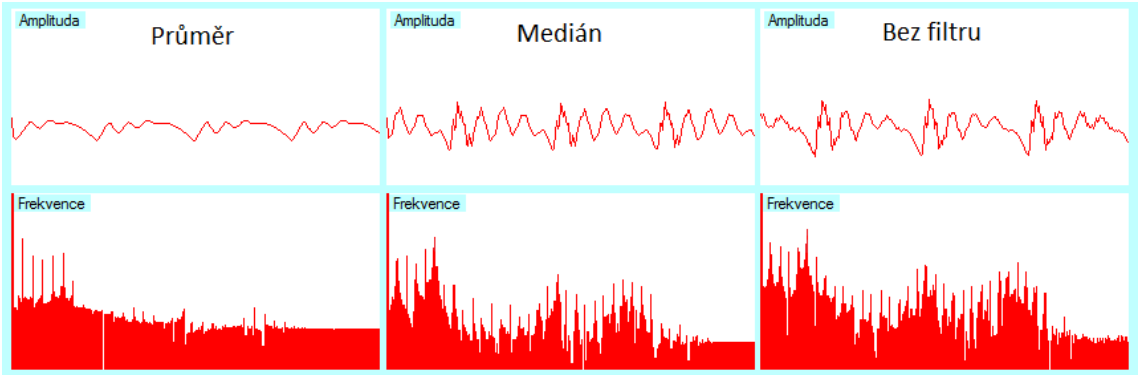
Funkce medián popsaná jednoduchý kódem přijímá data a pomocí rovnice k tomu určené eliminuje šumovou složku. Při této eliminaci omezí i užitný signál, ale omezení není tak rozdílné, aby se nedalo s daným signálem pracovat. Podle definice mediánu sečteme dva prvky pole, které vydělíme. V podstatě se jedná o průměr dvou prvků.

```
if (h >= 1){
     w = (_wavewave[h]+_wavewave[h-1])/2;
}
```

Průměrová elemininace šumu je velmi podobná s mediánem, že je zprůměrování hodnot nastaveno na výpočet s deseti prvky. Index definovaný prvek pole je v každém dalším o jeden menší z důvodu dosažení průměru posledních deseti prvků. Standardně probíhající smyčka připočítává proměnnou *h*, která má velikost délky pole.

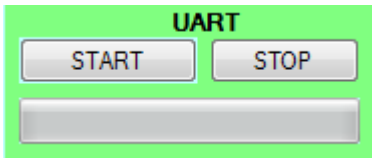
```
w = (_wavewave[h] + _wavewave[h - 1] +...+ _wavewave[h - 9]) / 10;
```

Na obrázku níže je názorná ukázka eliminace šumu pomocí funkcí průměru a mediánu. Graf vpravo zobrazuje původní signál. Amplitudové a frekvenční spektrum odpovídá vyslovenému písmenu **[e].** Mediánem snížíme všechny hodnoty o určité procento, ale jak lze vidět v grafu hodnoty zůstali poměrově zachované. U průměrové funkce to říci nemůžeme. Ačkoli je užitný signál stále použitelný, pro frekvenční analýzu bylo potlačeno příliš mnoho potřebných dat. Z tohoto důvodu může správně probíhat analýza jen u hodnot z analogového spektra.



Obrázek 26 - Zobrazení rozdílů filtrací

Blok pro spuštění komunikace mezi počítačem a MCU na vyrobeném HW umožnuje zapínání a vypínání čtení dat z komunikace UART.



Obrázek 27 – Panel pro spuštění komunikace UART

Po stisknutí tlačítka se provede následující kód. Nejdříve musí být nadefinován port, ze kterého se mají data číst. Jako další můžeme nastavit komunikační rychlost (Baud Rate). Nastavování rychlosti není povinné. Když ji nenastavíme, v systému se naprogramuje automaticky sama na standart 9600. Nastavení je doporučené z důvodu zajištění správné komunikace mezi zařízeními. Pokud je na jednom z nich rozdílná rychlost, potom přichází nepoužitelná data. Když je v definovaném portu zapojeno zařízení komunikující po UART projdeme přes ověřovací podmínku. Po dokončení spojení se spustí časovač, ve kterém se čtou data, jež dále můžeme vypisovat, zobrazovat v grafech nebo jakkoli dále zpracovávat.

```
serialPort1.PortName = "COM4";
serialPort1.BaudRate = 9600;   //nastavení komunikační rychlosti
serialPort1.Open();
if (serialPort1.IsOpen)
{
        timer1.Enabled = true;
        listBox1.Items.Add("UART_Zap");
        Start_UART.Enabled = false;
        Stop_UART.Enabled = true;
}
...
serialPort1.Read(buff, 0, 1);   //cteni ze serioveho portu
buffer_uart[w] = buff[0];
```

Kód pro odesílání dat zpět na HW zařízení funguje automaticky v závislosti na zpětných vazbách analýz řeči. Přijde-li zpětná vazba s textovým řetězcem tato data jsou uložena do pole typu *char* a jsou odeslána popsaným kódem do zařízení. Otevření portu je stejné jako při čtení, jen je rozdíl v zapisování.

```
serialPort1.Write(TextZpet, 0, 1);   //zapis do serioveho portu
```

Po zaslání dat celý proces analýzy řeči v programu Visual studia končí a čeká na nová data do další analýzy. Další operace po odeslání zpětné vazby po UART do HW zařízení jsou popsány níže v implementaci programu do HW.

4.3. Implementace programu na reálný HW

Veškeré popsané metody v předešlém programu byly zároveň implementovány do programu procesoru na DPS. Některé části kódu použité pro programovací jazyk .Net Framework (C#) musely být předělány na základní kód C. Díky psaní aplikace co nejjednodušším způsobem bylo konvertování dílčích částí kódu poměrně snadné.

4.3.1. Nahrávání a analyzování dat z mikrofonu

Nahrávání dat z mikrofonů je odlišné od nahrávání dat z mikrofonů v PC. Nejdříve se musí nastavit periférie ADC v procesor expertu viz. Obrázek 28.

Properties | Methods | Events

Name	Value	Details
A/D converter	ADC	ADC
Sharing	Disabled	
Interrupt service/event	Enabled	
A/D interrupt priority	medium priority	level 2, priority within level 3
A/D channels	2	
Channel0		
A/D channel (pin)	PTB0_TPM3CH0_AD1P0	PTB0_TPM3CH0_AD1P0
Channel1		
A/D channel (pin)	PTB1_TPM3CH1_AD1P1	PTB1_TPM3CH1_AD1P1
A/D resolution	Autoselect	12 bits
Conversion time	22.7 µs	21.935 µs
Low-power mode	Disabled	
Sample time	short	Total conv. time: high: 23.12 us
Initialization		
Enabled in init. code	yes	

Obrázek 28 - Nastavení ADC v procesor expertu

Pro použití dvou mikrofonů připojených přes zesilovače do procesoru nastavíme dva kanály pro ADC převodník. Na každý kanál definujeme jeden vstup do procesoru. Pro konverzi hodnot ve vzorkovací frekvenci 22 kHz nastavíme „Conversion time“ na 5,75µs. Tato hodnota byla vypočítána podle vzorce T=1/f. Menší čas pro konverzi nelze nastavit z důvodu kmitočtu procesoru definované externím krystalem. Pro analýzu není nutné snímat data v nejvyšší vzorkovací frekvenci. Po nastavení těchto dat byl napsán kód na vyčítání dat z ADC, která jsou dále předávána komunikačnímu protokolu UART pro analýzu slov nebo FFT. Po přepočtu pomocí FFT jsou analyzována data jako v programu pro použití ve Windows, např. hledání tónů stupnice, určování mužského nebo ženského mluvčího, a navíc je tento kód obohacen o metodu hledání zdroje zvuku. Následující kód popisuje vyčítání dat z ADC.

```
(void)AD1_MeasureChan(TRUE,1);      //povoleni konverze dat z ADC
(void)AD1_GetChanValue16(1,&valueP);//ulozeni dat z adc do promenne
vOutP[b]=(unsigned char) valueP; //předání dat do pole pro analýzu
```

4.3.2. Komunikace po sběrnici UART

Veškerá komunikace mezi počítačem a deskou probíhá skrz sběrnici UART. Není potřeba vytvářet složitý převodník, stačí pomocí čipu na desce určený pro konverzi dat na tento typ komunikace a u PC jen připojit kabel do příslušného portu USB. Pro příjem a zasílání dat je třeba nadefinovat asynchronní komunikaci viz Obrázek 29.

Properties | Methods | Events

Name	Value	Details
Settings		
Parity	none	none
Width	8 bits	8 bits
Stop bit	1	1
SCI output mode	Normal	
Receiver	Enabled	
RxD	PTE1_RGPIO1_RxD1	PTE1_RGPIO1_RxD1
RxD pin signal		
Transmitter	Enabled	
TxD	PTE0_RGPIO0_TxD1	PTE0_RGPIO0_TxD1
TxD pin signal		
Baud rate	262144 baud	262144 baud
Break signal	Disabled	
Wakeup condition	Idle line wakeup	
Transmitter output	Not inverted	
Receiver input	Not inverted	

Obrázek 29 - Nastavení sběrnice UART

Pro nastavení správné komunikace nejdříve musíme definovat vstupní a výstupní pin pro UART. Poté musíme navolit komunikační rychlost sběrnice. Byla nastavena komunikační rychlost na 262144 baud. Tato hodnota je maximální povolená hodnota sběrnice. Udává kolik hodnot za jednu sekundu, pošle po sběrnici.

```
for(a=0; a<257;a++)
{
        data3[1] = (unsigned char)vOutP[a];
        PutChar(data3[1]);
}
void PutString (const char* str) {
        while (str[0] != '\0') {
              PutChar(str[0]);
              str++;
        }
}
```

Pomocí výše uvedeného kódu plníme pole typu *char*, které dále předáváme metodě *PutString*. Tato metoda funguje jako delegát pro zasílání dat po komunikaci UART jinému zařízení. V našem případě do PC. Pomocí metody *RcvString* naopak zase získáváme data přijatá z PC. Tato data stejným způsobem vyčítáme, jak jsou ukládána. Dále je předáváme pro další zpracování.

4.3.3. Komunikace po sběrnici CAN

Jakákoli komunikace mezi jednotlivými deskami je prováděna po sběrnici CAN. Robotické zařízení (hlava) obsahuje tři univerzální desky. Ty analyzují a pracují se zvukem, obrazem nebo pohybem hlavy pomocí servo pohonů, dále Tower od společnosti Freescale vytváří komunikace mezi deskami a řídí jejich ovládání.

Princip funkce:

Na univerzální desku s připojenou přídavnou deskou na zvuk je zaslán požadavek na zjištění strany, ze které vychází nejsilnější zvuk.

```
mech_data.mouth.position.horizontal = 200;
```

Algoritmus vypočte souřadnice X, které jsou dále předány po sběrnici desce ovládající pohyb hlavy, následujícím kódem.

```
AN1_SendFrame(0, CAN_MSG_ID_POSITION, DATA_FRAME, 8, data);
```

Po zaslání požadavku změny pozice hlavy, se otočí na požadované souřadnice. Po té můžeme zaslat další požadavek na změnu polohy nebo necháme hlavu v dané pozici. Orientace pomocí zvuku je relativní a hlava se natočí přibližně středem ke zdroji zvuku. Deska s kamerami pak dále dohledá přesné souřadnice obličeje a polohu hlavy vycentruje na správný bod vypočítaných souřadnic.

```
data[3] = (uint8_t)mech_data.head.position.horizontal;
```

Pro další nastavování směrů hlavy a očí slouží následující příkazy, které ale v analýze zvuku nevyužíváme.

```
data[1] = (uint8_t)mech_data.eye.position.horizontal;
data[2] = (uint8_t)mech_data.eye.position.vertical;
data[4] = (uint8_t)mech_data.head.position.vertical;
```

Použitý kód popisuje pouze ovládání servo motorků, ale pomocí komunikace CAN se ovládá celá hlava. Tedy kromě desky na zpracování zvuku, také deska pro zpracování obrazu a pohonu. Jsou zde také zasílána data různých zpětných vazeb nebo požadovaných signálů pro vizualizaci na jiném místě. Celý kód je uložen v příloze s náležitými komentáři.

4.3.4. Mluvení robotického zařízení

Stejně jako u ADC, kde je převáděn analogový signál na digitální podobu, také u generování výstupního signálu musíme převést digitální signál na analogový. Tento způsob může být proveden přes složité zapojení digitálně analogových převodníků, u kterých bude dosaženo podobně kvalitního signálu, jako byl vstupní. Druhou možností, která byla implementována, je pomocí PWM signálu a zapojení dolnopropustného filtru.

Popis metody

Pro zprovoznění mluvení na reproduktor připojený k výstupu z MCU bylo nejdříve potřeba nastavit výstup na PWM signál viz obrázek níže.

Properties | Methods | Events

Name	Value	Details
PWM or PPG device	FTM22	FTM22
Duty compare		
Output pin	PTH0_FTM2CH2_AD1P20	PTH0_FTM2CH2_AD1P20
Interrupt service/event	Disabled	
Period	22.67 µs	0.023 ms
Starting pulse width	22.67 µs	0.023 ms
Aligned	Edge	
Initialization		
Enabled in init. code	yes	

Obrázek 30 - Nastavení výstupu na PWM

Na výstup je nastavena perioda 22,67µs pro dosažení vzorkovací frekvence 44,1kHz. Tato rychlost je stejná jako při načítání dat z mikrofonu. Pro definici výstupního signálu byla tudíž použita data přímo z mikrofonů. Tato data byla následně analyzována a podle nich vytvořena pole, která definují jednotlivá písmena mluvené abecedy. Pro poslání dat v podobě PWM slouží následující kód, který zrovna definuje výstup vyřčeného písmena „J“

```
for(jj=0;jj<241;jj++)
{
        PWM1_SetRatio16(JJJ[jj]);
}
```

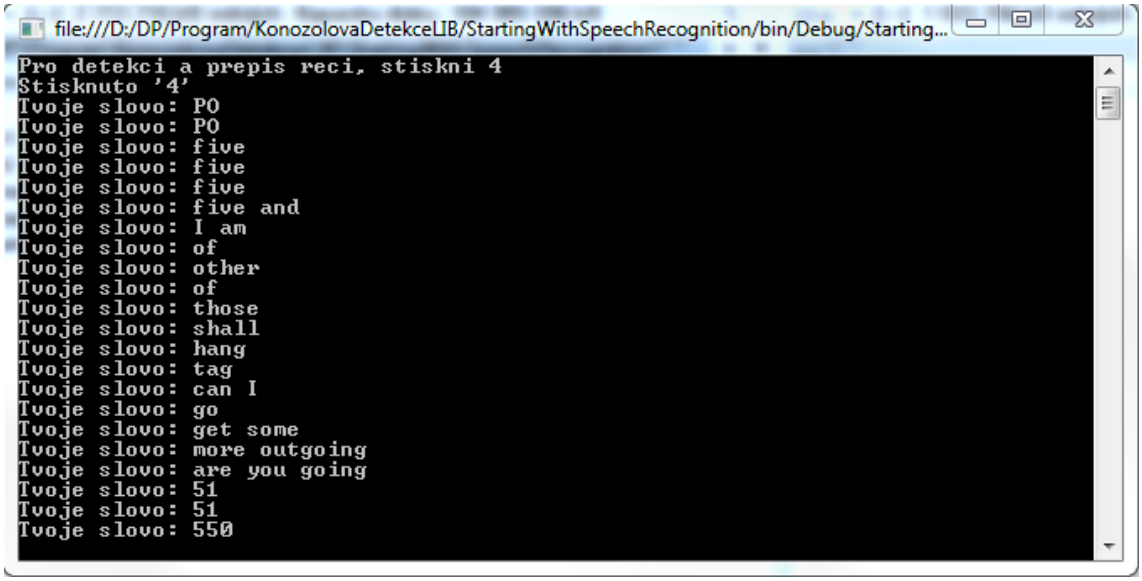
Kromě smyčky *for* jsou nejdříve pomocí příkazů *switch* a *case* vybrána písmena pro přehrání. Tímto způsobem můžeme nadefinovat jakékoli slovo a při požadované operaci je poskládáno a přehráno téměř jako jeden celek. V příloze je celý kód a v testování je popsán výsledek, jakým způsobem zařízení reprezentuje řeč.

5. Praktické ověření a testování robotického zařízení a jednotlivých implementovaných metod.

Touto kapitolou byly prakticky ověřeny funkce a metody popsané v této diplomové práci. Byly otestovány rozhodovací algoritmy detekce zvuku v programu pro PC, ale také i na univerzální desce.

5.1. Testování programu v PC pro analýzu anglického jazyka

V první řadě byl testován program, který byl vytvořen v konzolové aplikaci. Tento program byl vytvořen jen pro detekci řeči a jen v anglickém jazyce pomocí integrované knihovny pro Windows.



Obrázek 31 - Testování knihovny pro Windows

Na obrázku 31 je zobrazen výstup z knihovny při analyzování řeči. Tento způsob byl velmi nepraktický z důvodu pomalé analýzy, a také velkých nepřesností při analýze. Vyřčená slova musí být správně artikulována, jinak jsou analyzována špatně nebo vůbec. Dalším poznatkem je to, že při vyřčení delší věty nebo více slov za sebou, analýza trvá déle a výstup neodpovídá skutečnosti.

V tabulce níže jsou popsány časy trvání analýzy a rozdíl mezi slovy vyřčenými a vrácenou hodnotou.

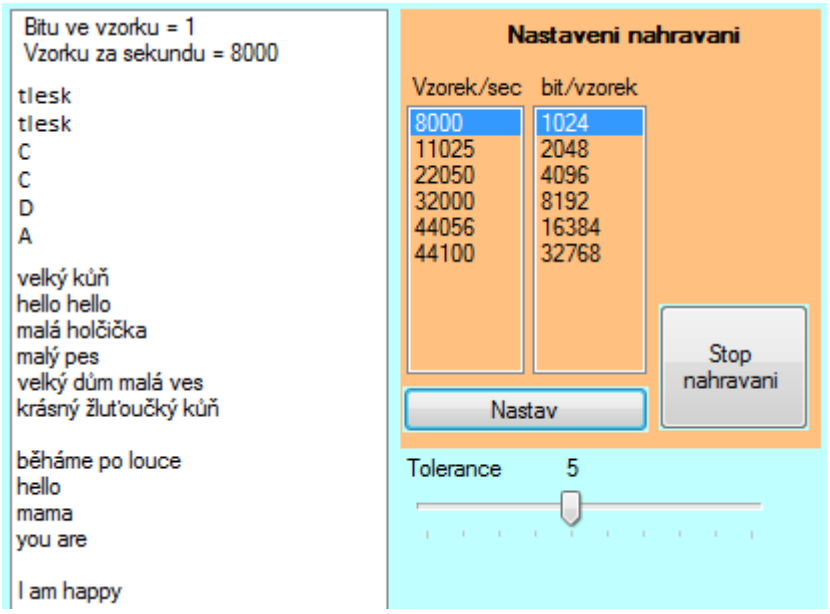
Vyřčené slovo	Vrácená hodnota	Doba analýzy [s]	Vyřčené slovo	Vrácená hodnota	Doba analýzy [s]
Hello	PO	4	choose	Those	5
Hello	PO	3	schell	Shall	2
Hi	five	6	hand	Hang	4
Five	five	4	Tag name	Tag	6
Hight five	five	5	Can I am	Can I	7
Five and six	Five and	6	Go	Go	2
I am	I am	3	Go home	Get some	6
On	Of	6	Are you going	More outgiong	8
Other	Other	4	Are you going	Are you going	10
Off	Of	5	fifty-one	51	5

Tabulka 5 - Testování knihovny pro Windows

Zelené zvýraznění znamená správnou analýzu řeči, modré značí částečně správnou nebo neúplnou analýzu a červené kolonky znamenají nesprávnou analýzu.

5.2. Testování hlavního programu pro PC

Program v PC funguje dvěma způsoby, jeden je probíhající analýza dat z mikrofonu počítače, druhý jako spojnice mezi programem v MCU na univerzální desce a internetem. Zároveň také slouží pro zobrazení dat posílané z MCU po sběrnici UART.



Obrázek 32 - Zpětné vazby po analýze

Po spuštění programu byla nastaveny data pro nahrávání a nejdříve byla otestována interní část pro analýzu. Výstupem této části byla data v podobě analýzy tónů stupnice C dur a hledání maxima viz tabulka níže.

zvuk	Vrácená hodnota	Doba analýzy [s]
Tlesk	Tlesk	1
Tlesk	Tlesk	2
Tón C	Tón C	2
Tón B	Tón C	3
Tón D	Tón D	2
Tón A	Tón A	3

Tabulka 6 - Detekce tónů

Při analýze neproběhla správně jen analýza tónu B ze stupnice C dur. K nesprávné analýze došlo z důvodu zvýšeného hluku v okolí a signál byl rušen. Dá se říct, že analýza hledání maxim a tónů probíhá na 80-95 % správně.

Další analýza probíhala po přepnutí na analýzu řeči ve spolupráci s Googlem. Při této analýze byly vyzkoušeny dva jazyky, čeština a angličtina. Tabulka níže zobrazuje výsledky analýzy.

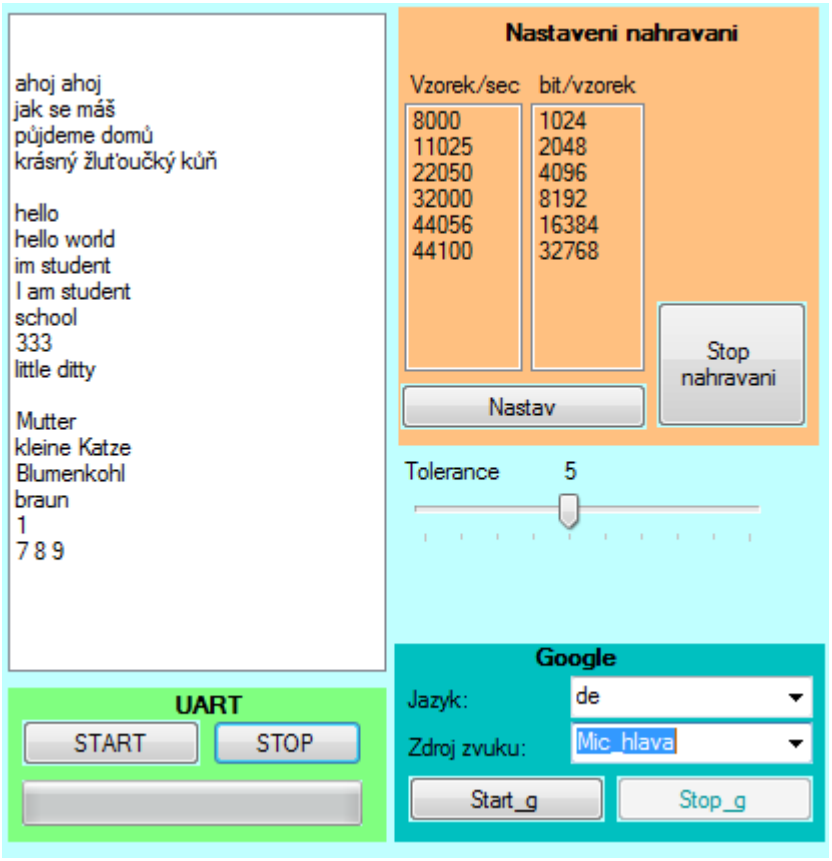
Čeština			Angličtina		
Vyřčené slovo	Vrácená hodnota	Doba analýzy [s]	Vyřčené slovo	Vrácená hodnota	Doba analýzy [s]
Velký kůň	Velký kůň	2	Hello	Hello	2
Helou helou	Hello hello	3	Mama	Mama	2
Malá holčička	Malá holčička	2	You are	You are	3
Malý pes	Malý pes	3	kitten		1
Velký dům malá ves	Velký dům malá ves	3	I am happy	I am happy	3
Krásný žluťoučký kůň	Krásný žluťoučký kůň	4	snail		1
Běháme po louce	Běháme po louce	2	castle		1

Tabulka 7 - Testování analýzy řeči pomocí Google

V analýze českých výrazů byl použit i anglický, modře označený, přepis proběhl správně, ale při rozhodování v jaké to bylo jazyce, není to správně. Při analýze v anglickém jazyce se vyskytly tři chyby. Po nahrání a poslání byla vrácena prázdná odpověď, tudíž vycházíme z faktu špatné výslovnosti slov. Rychlost analýzy dat spočívá v rychlosti připojení k internetu. Po odeslání jsou data téměř okamžitě analyzována. Prodleva mezi odchodem a návratem zpětné vazby je v závislosti na rychlosti poslání a přijetí dat.

5.3. Testování programu analýzy v procesoru na univerzální desce

Program v MCU byl testován pomocí připojené komunikace UART pro zobrazování dat zpětné vazby po příslušné analýze. Funkce pro zasílání dat zpět do procesoru nemohla být otestována z důvodu špatného optického členu pro galvanické oddělení mezi procesorem a komunikační sběrnicí UART. Tento krok byl vyměněn stiskem tlačítka pro nahrazení příkazu, na který se běžně čeká z komunikace.



Obrázek 33 - Zpětné vazby po analýze z MCU

Po připojení univerzální desky byl v programu vybrán zdroj, ze kterého mikrofonu byly brány data. Po spuštění komunikace UART a stisknutí tlačítka „Start_g“ byly nahrávána data a zasílány na server společnosti Google pro analýzu. V textovém poli vlevo jsou vidět výstupy zpětných vazeb. Testování bylo provedeno ve třech jazycích: čeština, angličtina a němčina. Analýza pomocí propojení HW s počítačem a následně internetem trvala o něco déle než analýza přímo z počítače. Časové prodlevy závisely na rychlosti připojení internetu, a také době odesílaných dat po sběrnici UART. V tabulce níže jsou data z uskutečněného testování.

Modré zbarvení tabulky označuje neúplný nebo částečně správný přepis, červené kolonky špatný přepis mluveného slova na text. Problém při testování se vyskytl jen v anglickém jazyce. Tento špatný přepis vznikl z důvodu špatné výslovnosti mluvčího.

Čeština			**Angličtina**		
Vyřčené slovo	**Vrácená hodnota**	**Doba analýzy [s]**	**Vyřčené slovo**	**Vrácená hodnota**	**Doba analýzy [s]**
Ahoj ahoj	Ahoj ahoj	6	Hello	Hello	5
Jak se máš	Jak se máš	5	Hello world	Hello world	6
Půjdeme domů	Půjdeme domů	7	I am student	Im student	6
Krásný žluťoučký kůň	Krásný žluťoučký kůň	8	I am student	I am student	7
Němčina			School	School	4
Mutter	Mutter	7	Little kitty	333	7
Kleine katze	Kleine katze	5	Little kitty	Little ditty	6
blumenkohl	blumenkohl	8			
braun	braun	9			
ein	1	5			
Sieben acht neun	7 8 9	6			

Tabulka 8 - Testování řeči na text z HW

Testování tónů hledání maxim a rozhodování jestli mluví muž a žena bylo provedeno pomocí manuálního určění stiskem tlačítka na desce. Tento způsob je dočasný z důvodu nefunkčního optického členu pro galvanické oddělení pro příjem příkazů z PC. Obrázen níže zobrazuje hodnoty zpětných vazeb od analýzy z procesoru. Pro tento způsob zobrazování hodnot stačí mít spuštěnou jen komunikaci mezi procesorem a PC. Pro vyfocení aktuálního stavu byl proces zastaven, aby byla data pohodlně zaznamenána v rámci testování.

Obrázek 34 - Testování tónů na desku HW

V obrázku výše při testování a odesílání dat z HW do PC vidíme, kdy byly poslány prázdné položky. Tyto „chyby“ jsou zobrazeny jako mezera mezi texty. Způsobená chyba vzniká při analyzování dat. Když je nahraný zvuk analyzován, ale není rovný žádnému z vzorových dat, je poslán prázdný příkaz. Tento způsob provedení je udělán proto, aby nebylo zbytečně zasíláno moc dat s popisem, že nebyl nalezen podobný signál. I když je tímto způsobem eliminován počet poslaných dat, tak i přes to program někdy pošle prázdný příkaz. V tabulce níže jsou popsána testovaná dat a jejich správnost po analýze.

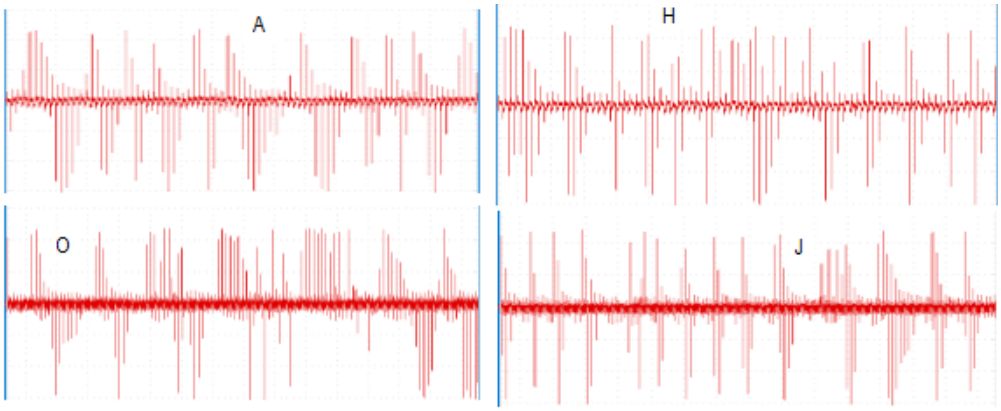
zvuk	Vrácená hodnota	Doba analýzy [s]	zvuk	Vrácená hodnota	Doba analýzy [s]
Tón B	Tón B	2	Tlesk	Tlesk	3
Tón C	Tón C	2	Tón B	Tón B	2
Tón E	Tón C	2	Tón E		3
Tón D	Tón D	3	Tlesk	Tlesk	2
Tón A	Tón A	2	Tlesk		2
Muž/žena					
muž	Muž	3	Žena	Žena	3
Muž	Muž	2	Muž	Muž	2
muž	žena	3	žena	žena	2

Tabulka 9 - Testování tónů a frekvencí lidské řeči

Testování a analýza probíhala téměř okamžitě, ale odezva při posílání dat pro zobrazení způsobovala delší prodlevu mezi vytvořeným zvukem, analyzováním a zobrazením na displeji PC. Chyby vznikaly u detekování tónu E, který buď byl přiřazen k tónu C, nebo vůbec k žádnému signálu. Při analýze muž/žena byl sporný jen jeden výsledek z šesti. A to díky zvýšené intonaci mluvícího muže. Podle reálných podmínek by jiný posluchač poznal rozdíl mezi mužem a ženou díky zabarvení hlasu a dalším složkám, které program v procesoru neanalyzuje. Vychází jen z počtu vyšších a nižších frekvencí obsažené v hlasu. Tento způsob je velmi nepřesný, ale jako základ rozhodování zcela postačuje.

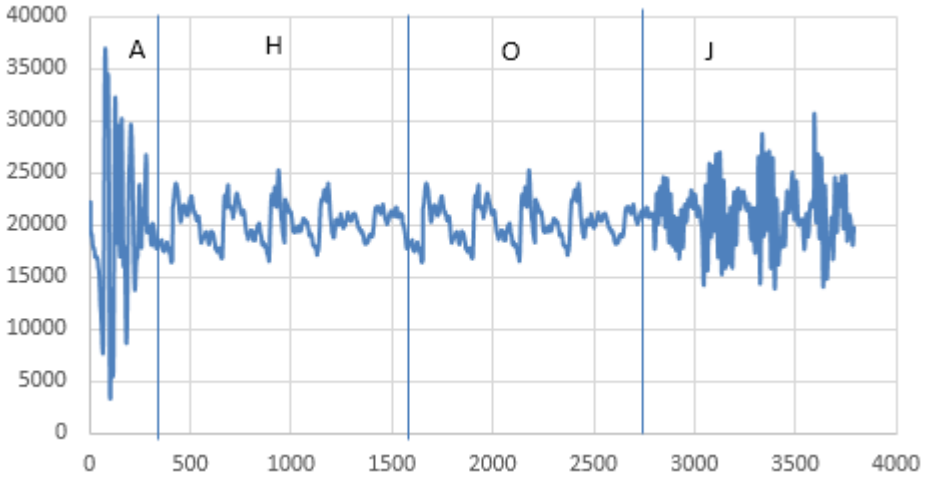
5.4. Testování mluvení z univerzální desky

V první řadě byla testována metoda na zasílání dat na reproduktor. Díky zapojení reproduktoru bez zesilovače výstupních dat je signál slabý a hůře detekovatelný. Pomocí měření na osciloskopu však byly ověřeny zaslaná data a tím i jejich správnost. Z již výše uvedeného důvodu nefunkčnosti optočlenu nemohli být zadány jakékoli příkazy pro mluvení. Na univerzální desce jsme nadefinovali několik příkazů pro písmenné spojení na utvoření slov. Pomocí tlačítka byla vyvolána metoda pro mluvení a následně jsme změřili výstup, abychom zjistili, zda odpovídá skutečnosti.



Obrázek 35 - Výstup z osciloskopu

Na výše zobrazených obrázcích jsou zvýrazněny jednotlivé části měřeného analogového spektra, které vytvářejí slovo „Ahoj“. Pro přesné zobrazení dat jsou jednotlivé části signálu přiblíženy natolik, aby byly zřetelné pro pozorování. Ve skutečnosti jsou spojeny v jeden celek utvářející dané slovo. Reálný tvar je zobrazen v grafu níže.



Obrázek 36 - Amplitudové spektrum slova "AHOJ"

V grafu jsou vyznačena jednotlivé písmena a jejich amplitudové spektrum. Jak je možné vidět, spektrum jednoho písmena se v reálném použití několikrát opakuje. To znamená, že šíření zvukových vln trvá nějakou dobu, což při nahrávání dat vypadá, jako by se jednotlivé písmeno zopakovalo mnohokrát za sebou, než je vyřčeno další. U některých písmen je znatelné zesílení a zeslabení signálu při přechodu na další vyřčené písmeno.

6. Závěr

Hlavním cílem této práce bylo vytvořit zařízení, které by reálně dokázalo analyzovat svoje okolí a pomocí detekce zvukových signálů rozeznávalo různé podněty a reagovalo na ně.

Teoretickými definicemi a rovnicemi je přiblížen náhled, jak funguje reálné zpracování signálů. Dále je zde popsáno rozdělení signálů, se kterými je možno pracovat, nakonec také charakter řečových signálů, ze kterých vycházíme v praktické části této práce.

Pro detekci a analýzu zvukových signálů je zapotřebí mít hardware pro program, který danou analýzu může provádět. Návrhem desek plošných spojů ve spolupráci s dalšími kolegy jsme vytvořili univerzální desku s procesorem a komunikačními periferiemi jak pro komunikaci mezi deskami pomocí sběrnice CAN, tak i mezi deskou a počítačem pomocí sběrnice UART. Na desku je zakomponován slot pro paměťovou kartu, na níž je možné ukládat data například statistiky nebo jako paměť pro poslané příkazy během funkčnosti komunikace. Jako další součást jsme přidali paměť RAM, již v této práci nevyužívám, stejně jako ukládání dat na SD kartu. Na desce s kamerovými senzory je však paměť RAM využívána skrze potřeby zvýšení dočasné operační paměti. Nepoužité vstupy a výstupy procesoru byly vyvedeny na patici, do které zapojujeme rozšiřující DPS. V našem případě se jedná o DPS se zapojením zesilovačů pro mikrofony a výstupem pro reproduktor.

V programu byly vytvořeny metody, díky kterým je možno analyzovat příchozí zvuková data do procesoru jak na univerzální desce, tak i do programu v PC. Nejdříve jsme při tvorbě metod navrhli nahrávání dat z připojeného mikrofonu do PC. Pro převod dat do frekvenčního spektra jsme napsali třídu pro rychlou Fourierovu transformaci. Problém této třídy byl z počátku v nastavení nahrávaných dat pro převod. Po vyřešení problému během vývoje programu mohly být napsány ostatní metody pro analýzy zvukových signálů.

Detekce jednotlivých tónů stupnice C dur je tvořena pomocí uložených dat do datových polí, a ty jsou porovnávány s aktuálně nahrávanými daty. Při testování tónů stupnice C dur byl použit program pro mobilní telefon s imitací klavíru. Shoda s reálným hudebním nástrojem je trochu rozdílná, hlavně barvou tónu, délky znění tónu. Hlavní porovnávanou složkou je však frekvence tónu, a ta je v porovnání stejná. Při testování v programu pomocí PC byla analýza o trochu přesnější než v procesoru, ale je možné říct, že s 80% šancí program dokáže určit správný tón či podnět.

Hlavní funkcí programu bylo vytvořit detekci mluvené řeči a její přepis do textové podoby. Původně měla být detekce vyřčených slov provedena podobně jako detekce tónů, ale jak se ukázalo během vývoje, byl tento způsob velmi nepraktický. Rozhodovací algoritmus by potřeboval velké množství uložených dat, což by extrémně zpomalilo jakékoli operace programu. V použití pro PC by nebyl tak velký problém, ale z důvodu následné implementace vytvořeného programu do MCU na univerzální DPS musí být program co nejméně náročný. Dalším návrhem řešení přepisu řeči do textu bylo zpracování pomocí knihovny pro Windows, ale jak se ukázalo, fungovala pouze pro anglický jazyk, navíc velmi nepřesně a pomalu. Posledním a finálním řešením, který nás napadl, je spojení se serverem společnosti Google. Jakékoli stránky, které využívají Google API, což je funkce pro přepis řeči na text, se spojují s touto funkcí bezplatně. V rámci vývoje a testování je tento způsob ověřený a naprosto spolehlivý.

Vytvořené metody pro program v PC byly náležitě upraveny tak, aby jejich funkce nebyla ovlivněna, ale fungovaly také pro MCU. Po implementaci a odladění programu byl nalezen hlavní problém, kvůli kterému nebylo možné analyzovat data z mikrofonů. Chyba nastala v nesprávném použití mikrofonního zesilovače. Jelikož napětí vycházející z mikrofonu je velmi malé, řádově do 10 milivoltů, a byl použit zesilovač, který zesiluje napětí od 300 milivoltů, tak jediný zesílený signál bylo rušivé napětí naindukované z okolí. Po použití tzn. před napětí vytvořeného na výstupech z mikrofonů, byl problém odstraněn. Signál byl sice velmi slabý oproti signálu použití z mikrofonu pro PC, ale to vychází ze závislosti na použitém HW, který na to není ideálně stavěn.

Poslední částí práce se zvukovými signály na desce plošného spoje byla úloha rozjetí mluvení pomocí reproduktoru. Díky správně propojenému výstupu podporující signál PWM se nám práce podstatně zjednodušila. Pomocí vytvořené metody měníme šířku pulzu PWM a tím dosáhneme signálu odpovídající analogovému spektru mluveného signálu. Pomocí RC článku je tento PWM signál upraven z digitální podoby obdélníkového signálu na analogovou podobu. Pomocí kondenzátoru, který se nabijí a vybíjí v závislosti na délce logické „1“, je tvořen potřebný signál. Tento signál prochází cívkou reproduktoru a tak mění elektrické pulzy na akustické vlny. Lidské ucho tyto signály vnímá jako vyřčená slova. V porovnání z lidskou mluvou je zvuk nesrovnatelný a velice nedokonalý. Pro imitaci lidského hlasu se používají různé syntetizátory, ty však u této aplikace nebyly použity.

. .

7. Literatura:

[1] <http://fyzika.fce.vutbr.cz>
http://fyzika.fce.vutbr.cz/file/martinek/frekvanalzvuku.pdf [Datum citace 3.10.2012]

[2] DOLEŽAL, Aleš. *Programové vybavení pro frekvenční analýzu zvukového signálu*. Zlín, 2006. Bakalářská práce. Univerzita Tomáše Bati ve Zlíně. [Datum citace 3.10.2012]

[3] < http://tvorbawebu.wz.cz >
http://tvorbawebu.wz.cz/faiirfe/help/vzorkovani.htm [Datum citace 4.10.2012]

[4] < http://www.ivokucera.com >
http://www.ivokucera.com/encyklopedie/personality.php?id=7332 [Datum citace 4.10.2012]

[5] KLÍČ, Alois, Miroslava DUBCOVÁ a Karel VOLKA. *Fourierova transformace*. Vyd. 3. Praha: VŠCHT, 2002, 196 s. ISBN 80-708-0478-5. [Datum citace 6.3.2014]

[6] KLÍČ, Alois. *Fourierova transformace*. 2. přepr.vyd. Praha: Vydavatelství VŠCHT, 1994, 178 s. ISBN 80-708-0345-2 [Datum citace 4.10.2012]

[7] < http://en.wikipedia.org >
http://en.wikipedia.org/wiki/Carl_Friedrich_Gauss [Datum citace 9.10.2012]

[8] <*http://cs.wikipedia.org*>*http://cs.wikipedia.org/wiki/Rychl%C3%A1_Fourierova_transformace* [Datum citace 10.10.2012]

[9] SIGMUND, Milan. *Rozpoznávání řečových signálů: přednášky*. 1. vyd. Brno: VUT FEKT, ústav radioelektroniky, 2007, 122 s. ISBN 978-80-214-3526-1.[Datum citace 10.8.2013]

[10] Shannon, C.E.: A Mathematical Theory of Communication. BSTJ, Vol.27, 1968, pp. 623-656[Datum citace 3.11.2013]

[11] Psutka,J.: Komunikace s počítačem mluvenou řečí. Academia, Praha,1995. [Datum citace 12.11.2013]

[12] <http://cache.freescale.com>*http://cache.freescale.com/files/32bit/doc/ref_manual/MCF51AC256RM.pdf* [Datum citace 24.11.2012]

[13] HRBÁČEK, Jiří. [i]Komunikace mikrokontroléru : s okolím 2.[/i] 1.vyd. Praha : BEN-technická literatura, 2000. 151 s. ISBN 80-86056-73-2.[Datu citace 6.3.2013]

[14] <http://cs.wikipedia.org/ > *http://cs.wikipedia.org/wiki/CAN_bus* [Datum citace 12.3.2014]

[15] < http://www.hep.upenn.edu> *http://www.hep.upenn.edu/SNO/daq/parts/tlc271.pdf*

[16] MAREŠ, Amadeo. [i]1001 Tipů a triků pro C#.[/i] 1.vyd. Praha : Computer Press, a.s., 2008. 360 s. ISBN 978-80-251-2125-2.[Datum citace 20.10.2012]

[17] MANN, Burkhard. [i]C pro mikrokontroléry : uC & praxe.[/i] 1. české vydání. Praha : BEN-technická literatura, 2003. 279 s. ISBN 80-7300-077-6. [Datum citace 20.10.2012]

[18] ROZEHNAL, Zdeněk. [i]Mikrokontroléry MOTOROLA : HC11.[/i] 1.vyd. Praha : BEN-technická literatura, 2001. 191 s. ISBN 80-86056-77-5.

[19] <http://www.fi.muni.cz/>*http://www.fi.muni.cz/usr/jkucera/pv109/2001/xuher/xuher.html* [Datum citace 20.3.2014]

[20] < http://stackoverflow.com >*http://stackoverflow.com/questions/8754111/how-to-read-the-data-in-a-wav-file-to-an-array* [Datum citace 23.10.2012]

[21] <https://developers.google.com >*https://developers.google.com/translate/v2/using_rest* [Datum citace 10.4.2014]